./

GENESIS - Learning Outcome & Mini-project Summary Report



|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Ver. Rel. No.** | **Release Date** | **Prepared. By** | **Reviewed By** | **To be Approved** | **Remarks/Revision Details** |
| 1.0 | 16-04-2021 | Sourav Dey  (99003785) |  |  | Addition of Mini projects of SDLC and Python. |
| 1.1 | 17-04-2021 | Sourav Dey  (99003785) |  |  | Addition of Mini projects of Embedded C and MBSE. |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |

**Details**

Contents

[Contents 3](#_Toc69477444)

[Miniproject -1 SDLC (Software Development Life Cycle)-> [Team] 4](#_Toc69477445)

[Modules Used: 4](#_Toc69477446)

[Topic and Subtopics: 4](#_Toc69477447)

[Objectives & Requirements 4](#_Toc69477448)

[Design 8](#_Toc69477449)

[Test Plan 10](#_Toc69477450)

[Implementation Summary 12](#_Toc69477451)

[Video Summary 13](#_Toc69477452)

[Git Link 13](#_Toc69477453)

[Git Dashboard 13](#_Toc69477454)

[Summary 13](#_Toc69477455)

[Individual Contribution & Highlights 19](#_Toc69477456)

[Contributor’s List 19](#_Toc69477457)

[Summary 19](#_Toc69477458)

[Challenges faced and how were they overcome 20](#_Toc69477459)

[Future Scope (If applicable) 20](#_Toc69477460)

[Miniproject -2 Python-> [Individual] 21](#_Toc69477461)

[Modules Used 21](#_Toc69477462)

[Topic and Subtopics 21](#_Toc69477463)

[Objectives & Requirements 21](#_Toc69477464)

[Design 22](#_Toc69477465)

[Implementation Summary 26](#_Toc69477466)

[Git Link 26](#_Toc69477467)

[Git Dashboard 26](#_Toc69477468)

[Summary 26](#_Toc69477469)

[Challenges faced and how were they overcome 26](#_Toc69477470)

[Miniproject -2 Python-> [Individual] 27](#_Toc69477471)

[Module/s 27](#_Toc69477472)

[Topic and Subtopics 27](#_Toc69477473)

[Objectives & Requirements 27](#_Toc69477474)

[Design 27](#_Toc69477475)

[Test Plan 27](#_Toc69477476)

[Implementation Summary 27](#_Toc69477477)

[Git Link 27](#_Toc69477478)

[Git Dashboard 27](#_Toc69477479)

[Summary 27](#_Toc69477480)

[Individual Contribution & Highlights 27](#_Toc69477481)

[Summary 27](#_Toc69477482)

[Challenges faced and how were they overcome 27](#_Toc69477483)

# Miniproject -1 SDLC (Software Development Life Cycle)-> [Team]

## **Modules Used:**

Modules used in this project are as follows:

* C
* SDLC

Project Title: Mini Calculator

### Topic and Subtopics:

1. The core steps of the SDLC is being implemented.
2. Features of the calculator is being implemented.
3. Testing is done against each function created.
4. Introduction about SDLC.
5. C Programming
6. Code Analysis.
7. CPP check
8. Valgrind
9. Testing
10. Unit Testing
11. Make file
12. V Model
13. Agile Model
14. Git Hub

## **Objectives & Requirements**

Our Product

Objectives:

1. The calculator will solve all the basic arithmetic operations like Addition, Subtraction, Multiplication, Division and Modulus.
2. The calculator will solve all exponential, logarithmic, power of a number, factor of a number operations.
3. The calculator will perform complex number operations.
4. The calculator will have a LCD to display the result and a small storage unit to store last five results.

SWOT Analysis:

Strength:

1. Dual Power source
2. Pocket Friendly
3. Memory storage
4. Accuracy and speed
5. Attractive

Weakness:

* Less number of complex functions.
* Display digits are limited.

Opportunities:

* Can be used by a wider age group.
* As education is an important aspect in India, it will have a greater market value.

Threats:

* Competitive market.
* Application of smartphones reduces the usage of calculators.
* Running ahead of the market.

4W&1H:

What: User friendly and pocket friendly Calculator.

Who: People of any age group can use the calculator as it is user friendly i.e. students in schools and colleges, Officer’s in industries, self- employees.

When: It is used for students for saving time and for Officer’s, employees for complex calculations (use Scientific calculator), physics formulae for students.

Why: For saving time.

How: By operating manually for analog calculators and using screen touch for smart calculators.

Requirements:

High Level Requirements:

1. Arithmetic Operations.
2. Trigonometric Operations
3. Logarithmic Operations
4. Cube Roots
5. Math print
6. Power of a number.
7. Memory Storage.
8. Binary to Decimal.
9. Complex Numbers.
10. Maximum input user can give is 12 digits.

Low Level Requirements:

| **Requirement** | **Description** |
| --- | --- |
| Binary to Decimal Conversion | Take the input in form of only 1s and 0s as long through keypad and accordingly give the output as an int. |
| Root and Power | Take input as a double and find the square root and give output in double type. To find power take inputs as int for both number and the |
| Arithmetic Operations |  |
| Addition | Input validation: check the ASCII value range of the user input numbers. Input type: integer, float. |
|  | Operation: Take two inputs from the user and check the data type. |
|  | If the inputs are in float data type; the results will be in floating point. If the inputs are in integer data type; |
|  | the result will be in integer data type. If the inputs are in combination of integer as well as floating type; |
|  | then the result should be in floating type. |
| Subtraction: | Input validation: check the ASCII value range of the user input numbers. Input type: integer, float. |
|  | Operation: Take two inputs from the user and check the data type. Sign of both the input values must be considered and accordingly the |
|  | result should be in floating type. |
| Multiplication: | Input validation: Check the ASCII value range of the user input numbers. Also check the sign of the user input numbers. |
|  | Input type: integer, float. Operation: Take two inputs from the user and check the data type. Sign of both the input values must be |
|  | considered and accordingly the multiplication operation must be performed. If the inputs are in float data type; the results will be in |
|  | floating point. If the inputs are in integer data type; the result will be in integer data type. If the inputs are in combination of integer |
|  | as well as floating type; then the result should be in floating type. |
| Division: | Input Validation: Check the ASCII value range of the user input numbers. Also check the sign of the user input numbers. |
|  | Divide by zero is not possible. Input type: integer, float. Operation: Take two inputs from the user and check the data type. |
|  | Sign of both the input values must be considered and accordingly the division operation must be performed. |
|  | If the inputs are in float data type; the results will be in floating point. If the inputs are in integer data type; the result will be in |
|  | are in combination of integer as well as floating type; then the result should be in floating type. |
| Memory Storage | A history button is created which shows the last five stored results. When the user hits the HISTORY button it will display the last five |
|  | stored value. Operation: Works with arrays. |
| Complex mode | It is used to calculate with the real and imaginary numbers in a single mode. |
| Math Print | It takes the input, calculates it and shows the result as well as the input together. |
| Trigonometric Functions | Finding values for sin (), cos (), tan(), sec(), cosec(), cot() functions. |
| Exponential functions | To perform logarithmic functions and cube root functions. |

## **Design**

HLR using UML Diagrams:

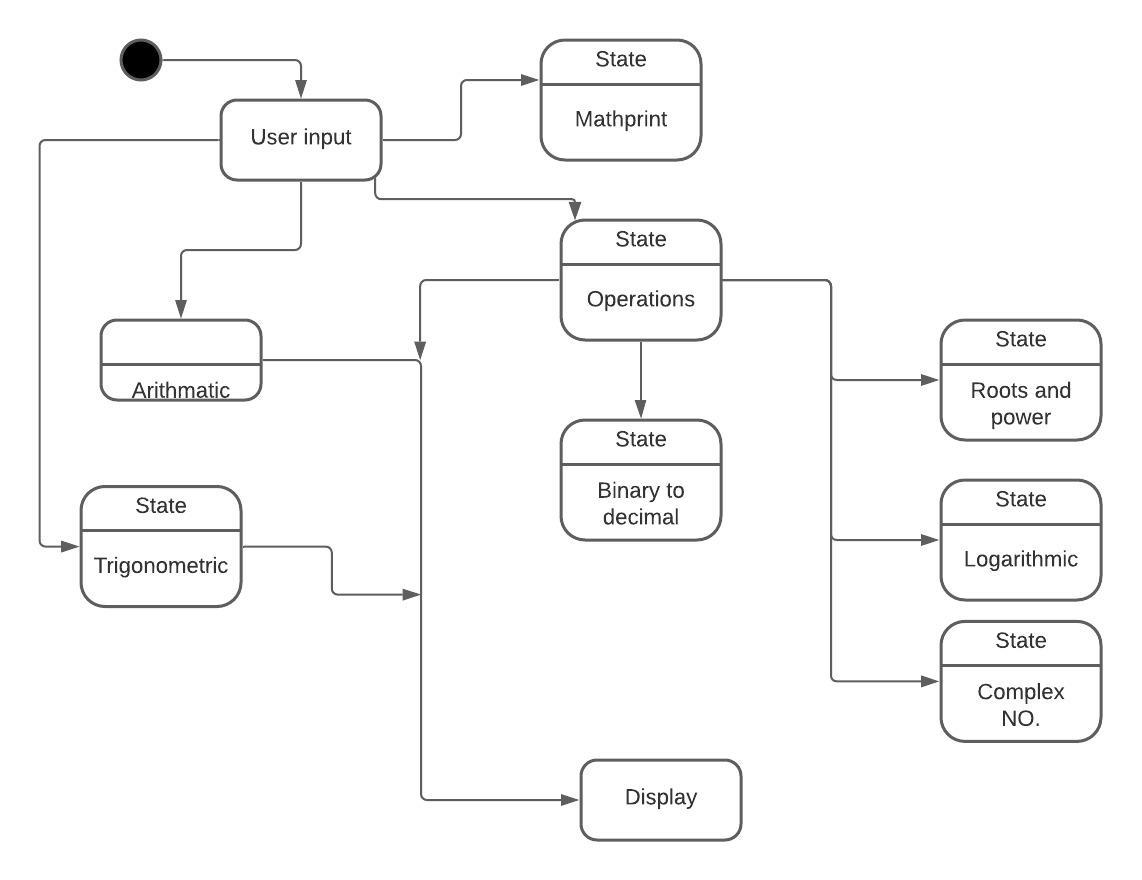


Figure : HLR using Activity diagram.

LLR using UML Diagrams:

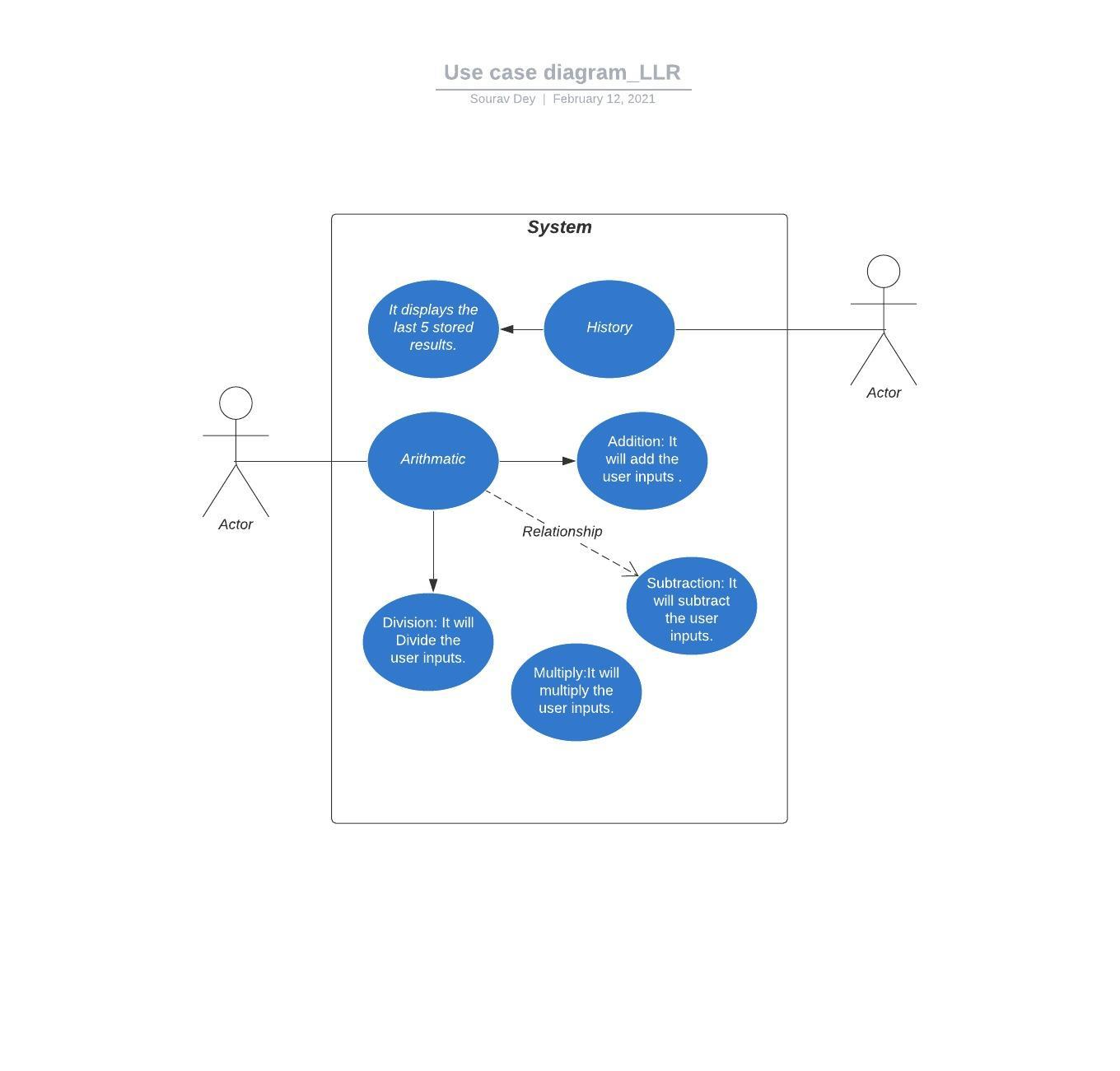


Figure : LLR using Use case diagram.

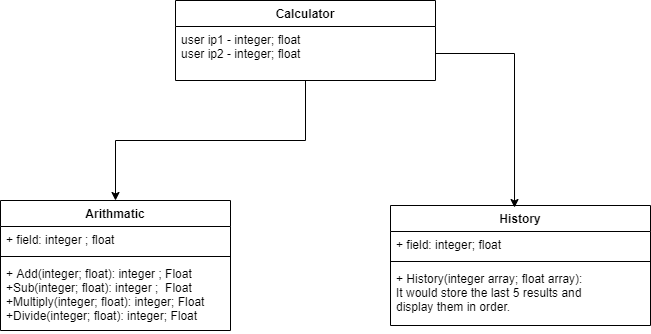


Figure : LLR using class diagram.

## **Test Plan**

| **Test\_id** | **Description** | **Expected input** | **Expected Output** |
| --- | --- | --- | --- |
| LLR\_1 \_ Arithmetic Operations | It contains all the basic arithmetic operations |  |  |
|  | Addition:1) The user input must be validated. The sign of the user input must also be validated. | Integer, integer | Integer, integer |
|  | 2) The floating-point input must provide a floating-point result. | Float, Float | Float |
|  | 3) A combination of floating point input and integer input must provide a floating-point output. | Float, integer Or Integer, float | Float |
|  | 4) If result exceeds by 14 digits | Input 1=10 digits Input 2=6 digits | Out of range |
|  | 5) If the first input is a negative number and second input is positive number or vice-versa. | Input1= -ve greater Input2= +ve smaller | Negative |
|  | If the negative input is greater than the positive input then the output must be negative. | Or Input1= +ve smaller Input2= -ve greater | Negative |
|  | Subtraction: 1) The user input must be validated. The sign of the user input must be validated. | Integer or floating-point input. | pass |
|  |  | Alphanumeric | Error |
|  | 2) If both the input is of integer type or floating type then the output must be integer or floating type. | Integer, integer | Integer |
|  |  | Or Float, float | Float |
|  | 3)If both the input sign is negative then the output must be the additive of both the values. | Input1= -ve Input2= -ve | Output=-(input1+input2) |
|  | 4)If the result exceeds 14 digits then the display unit must show out of bound or out of range | Input1=more than 14 digits | Output= result out |
|  | Multiplication:1) The user input must be validated. The sign of the user input must be validated. | Integer or floating-point input. | Pass |
|  |  | Alphanumeric input | Error |
|  | 2) If both the input is of integer type or floating type then the output must be integer or floating type. | Integer, integer | Integer |
|  |  | Or Float, float | float |
|  | 3)If both the values are negative the output must have a positive sign. | Input1=-ve Input2=-ve | Output=+ve |
|  | If one input is positive and other one is negative then the resultant must have negative sign. | Input1=+ve Input2=-ve | Output=-ve |
|  | Division:1) The user input must be validated. The sign of the user input must be validated. | Integer or floating-point input. | Pass |
|  |  | Alphanumeric input | Error |
|  | 2) If both the input is of integer type or floating type then the output must be integer or floating type. | Integer, integer | Integer |
|  |  | Or Float, float | float |
|  | 3) If both the values are negative the output must have a positive sign. | Input1=-ve Input2=-ve | Output=+ve |
|  | If one input is positive and other one is negative then the resultant must have negative sign. | Input1=+ve Input2=-ve | Output=-ve |
|  | 4) If the denominator is zero then the display unit must show error. | Input1 = digit Input2 = zero | Error |
|  | If the numerator is zero it must display infinite | Input1=zero Input2= digit | infinite |
| LLR\_2\_Memory\_Storage | 1)It must display the last five results when the user hits the history button. | History | Last five results |
|  | 2) The history operation starts storing the results from first after switching on the calculator. | OFF | No result |

## **Implementation Summary**

Implementation folder had all source files, header files, test files for different features of the calculator such as Basic Arithmetic, Square root, cube root, exponent, logarithm, etc.

Here, **inc** folder holds all the header files with “.h” extension which contains prototype of all functions, structure definition, macro definition and definition of all the enumerators.

The **src** folder holds all the source files with “.c” extension which has definitions of all the functions whose prototype is defined in header files.

The **test** folder holds the ***test\_calculator\_operations.c*** file for cumulative testing of the source codes based on requirements, scenario and boundary.

Other than these folders, there is also a **unity** folder which holds prototypes and definition of the standard unity test case functions.

Also, there is a **Make file** which builds, debugs using Valgrind, check static and dynamic code quality, performs overall unit testing for all the codes together with the execution of single commands based on different defined targets.

### Video Summary

“Please upload a short video on the repo for the walkthrough of the project (Team/Individual) less than 7min and less than 30MB File Size. Start is the Standard opening slide with title of miniproject + Team members followed by the walkthrough ”

### Git Link

<https://github.com/99003783/T7_SDLC_CALC>

### Git Dashboard

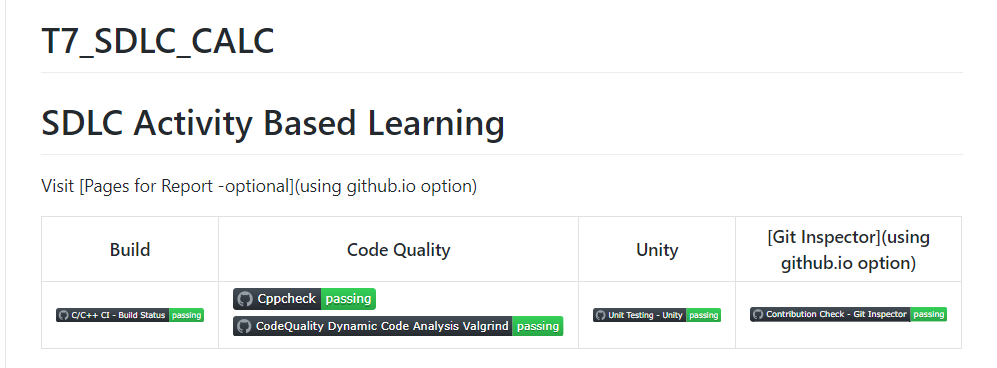


Figure : Git dashboard

### Summary

In this project, we mainly-focused on how to design a calculator which is bit different and cost-effective as compared to other calculators present in the market.

So, for this we first analyzed all the other calculators from low-end feature low cost calculator to high-end feature high-cost calculator and prepared a list of features to include in our modified cost-effective more featured calculator.

Features included in the calculator are basic arithmetic operations (addition, subtraction, multiplication, division, modulus), complex number operations, power of a number, trigonometric and inverse trigonometric functions, log and exponential functions and binary to decimal conversion.

This calculator is implemented through C programming. This calculator will be mainly used by the school, college students, scientists, businessman for various purpose. Also, this calculator will be cost effective and if implemented over hardware it cost around 1000 – 1500 INR.

#### Git inspector summary

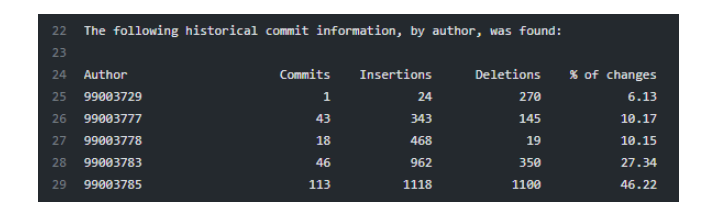


Figure : Git inspector

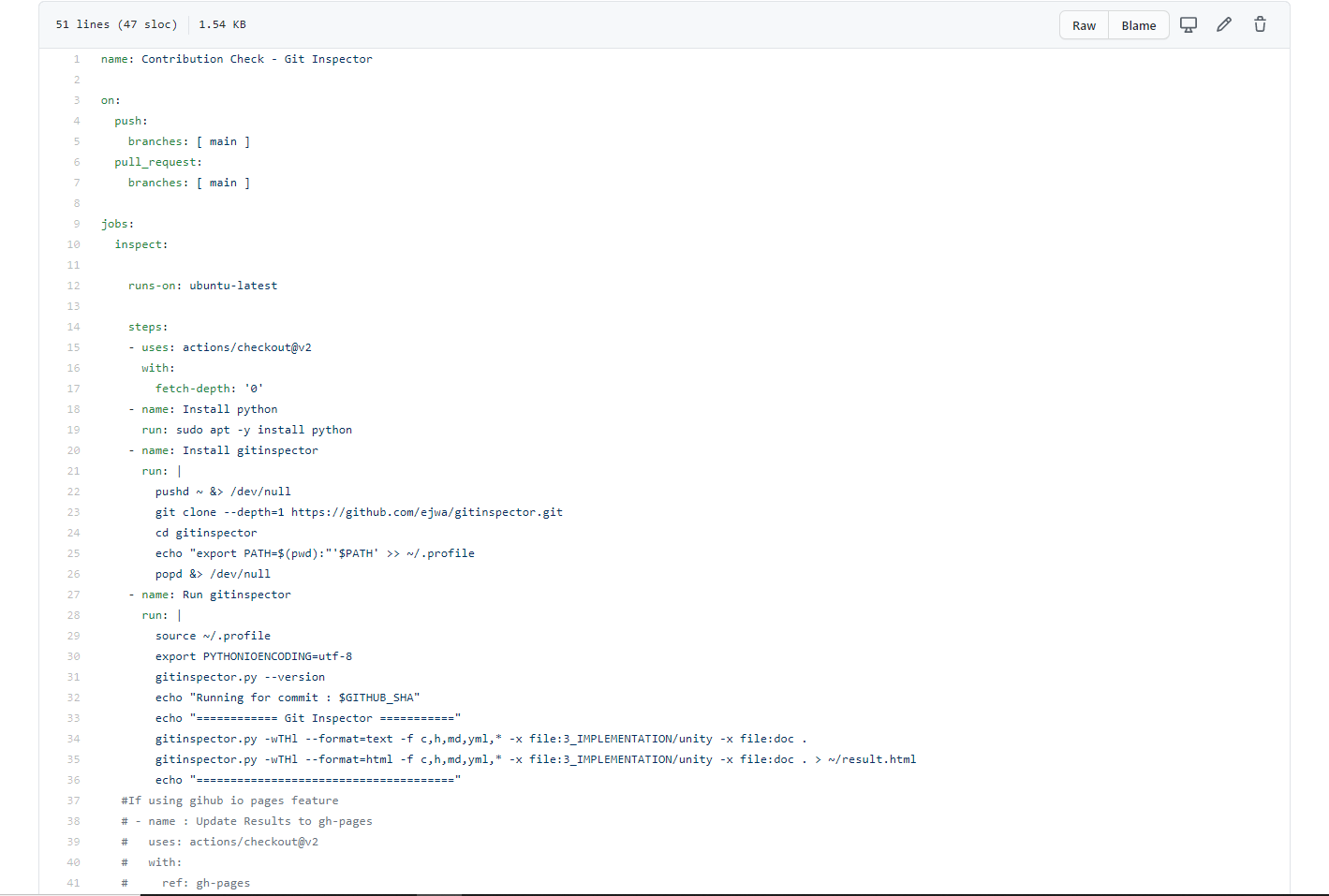


Figure : Implementation code for git inspector

#### Build

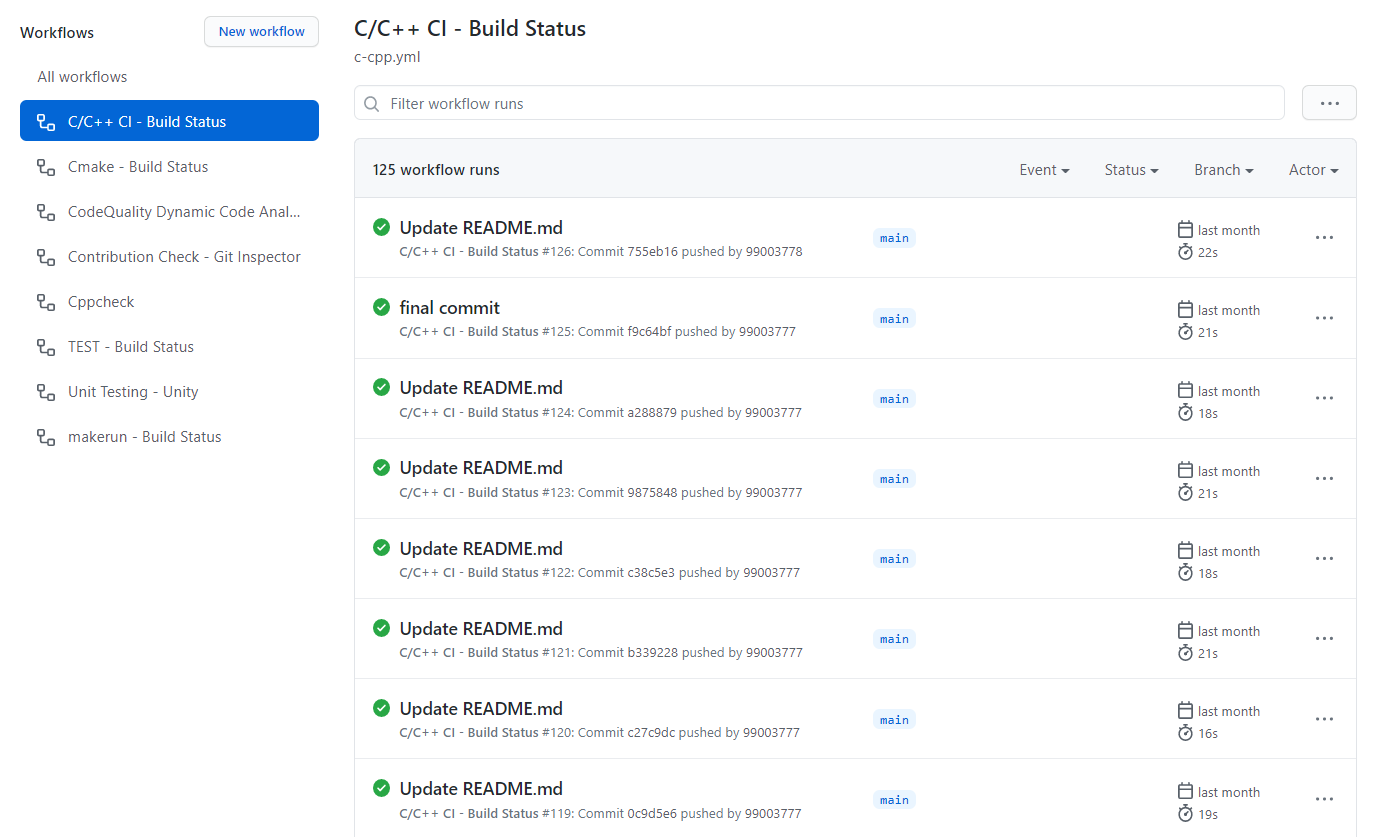


Figure : Build status

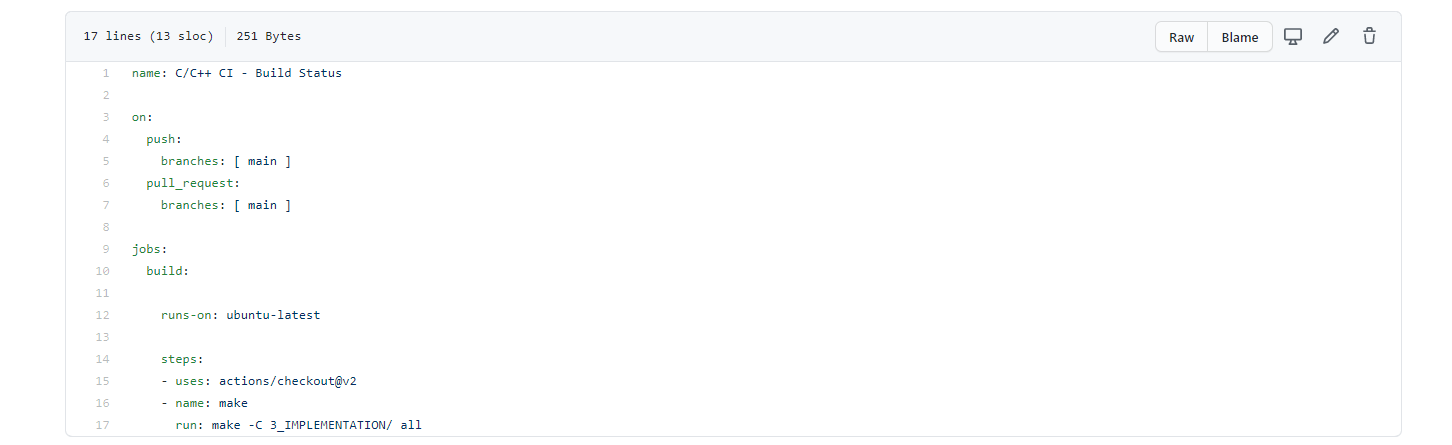


Figure : Workflow code for checking the build status

#### Code quality and Issues or Bug Tracking

Static Code Analysis:

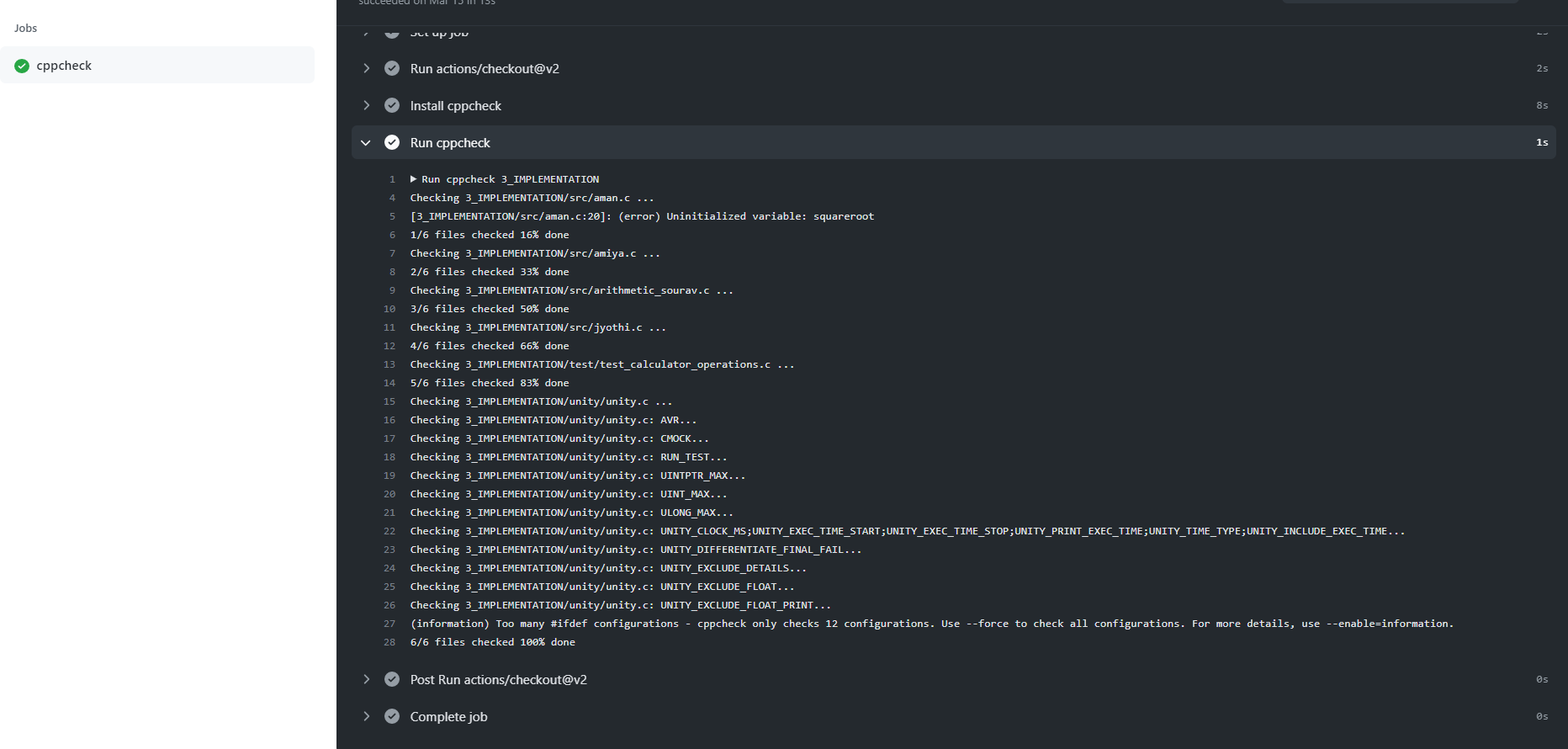


Figure : CPP check

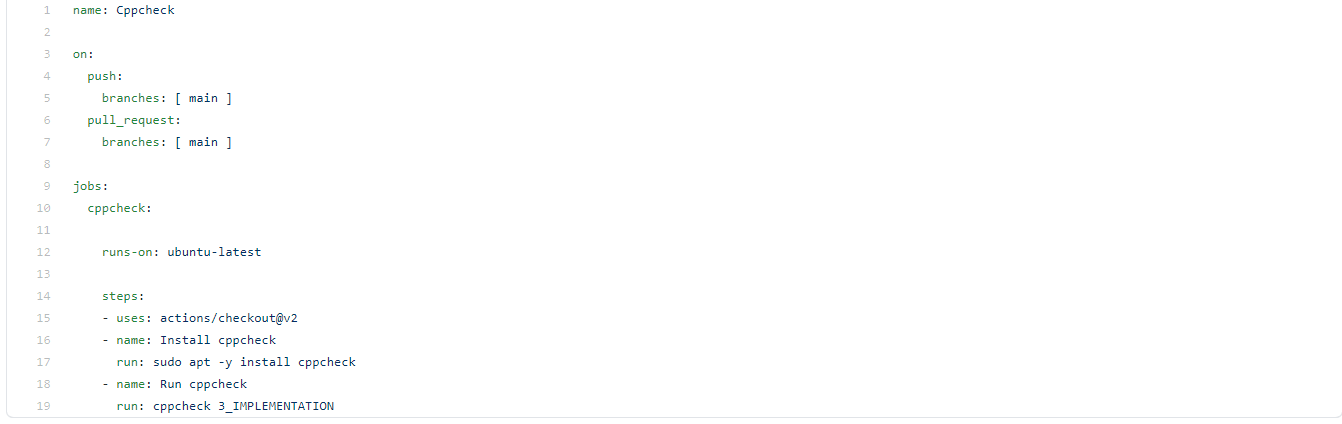


Figure : Code for Cpp check

Dynamic Code analysis:

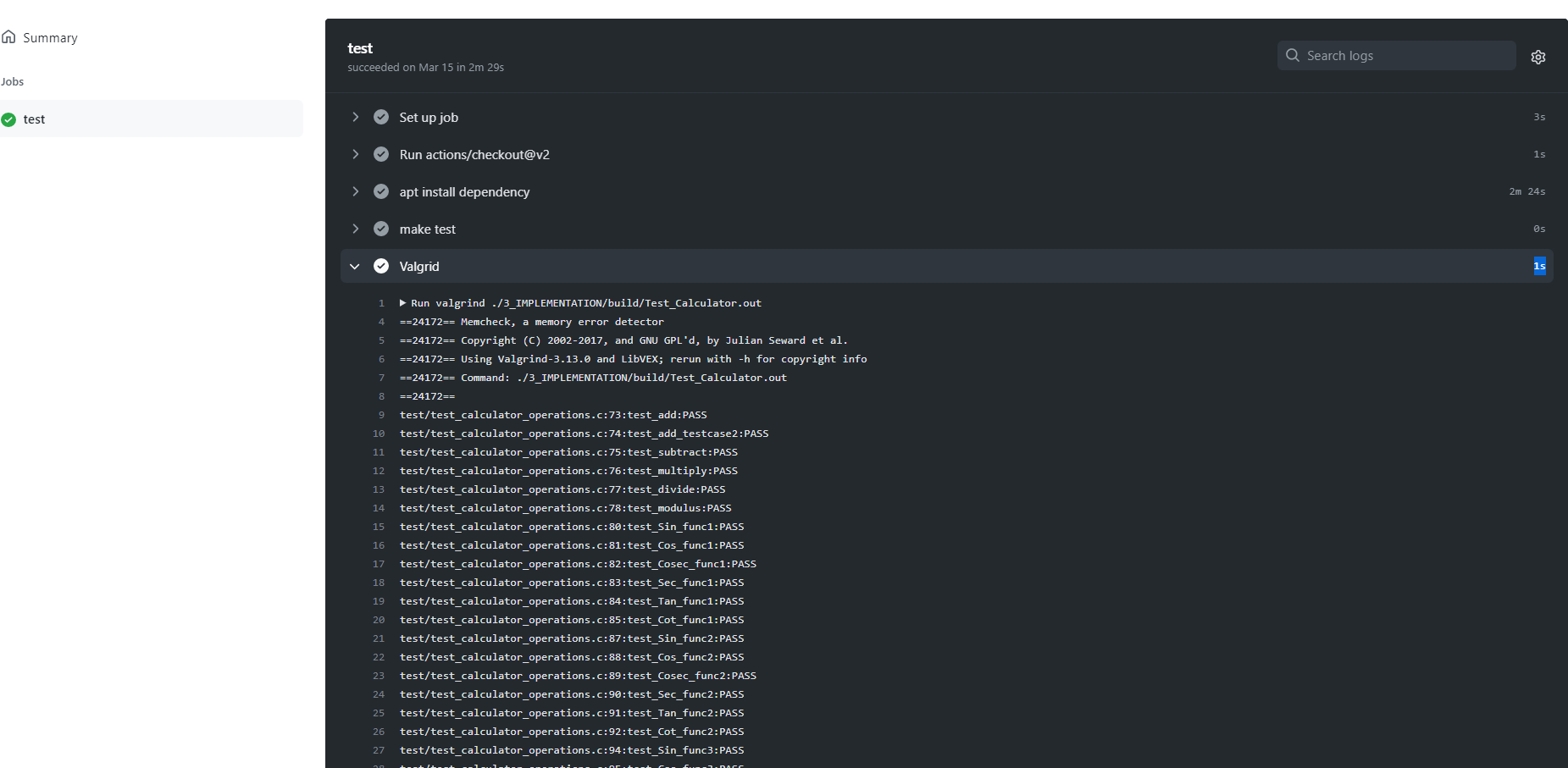


Figure : Dynamic Code Analysis

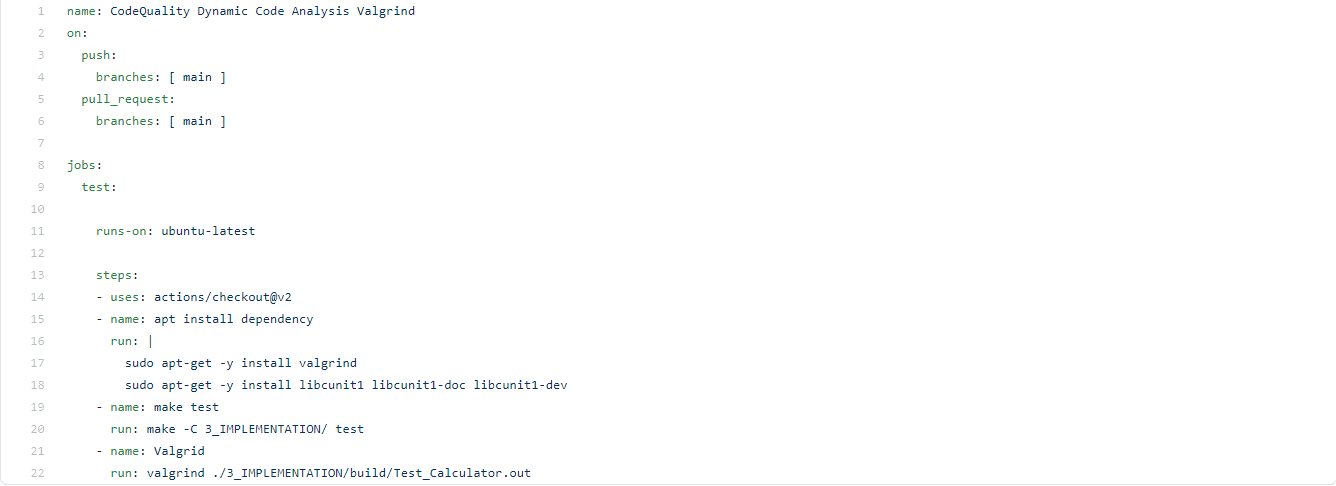


Figure : Valgrind check code

Git Issues:

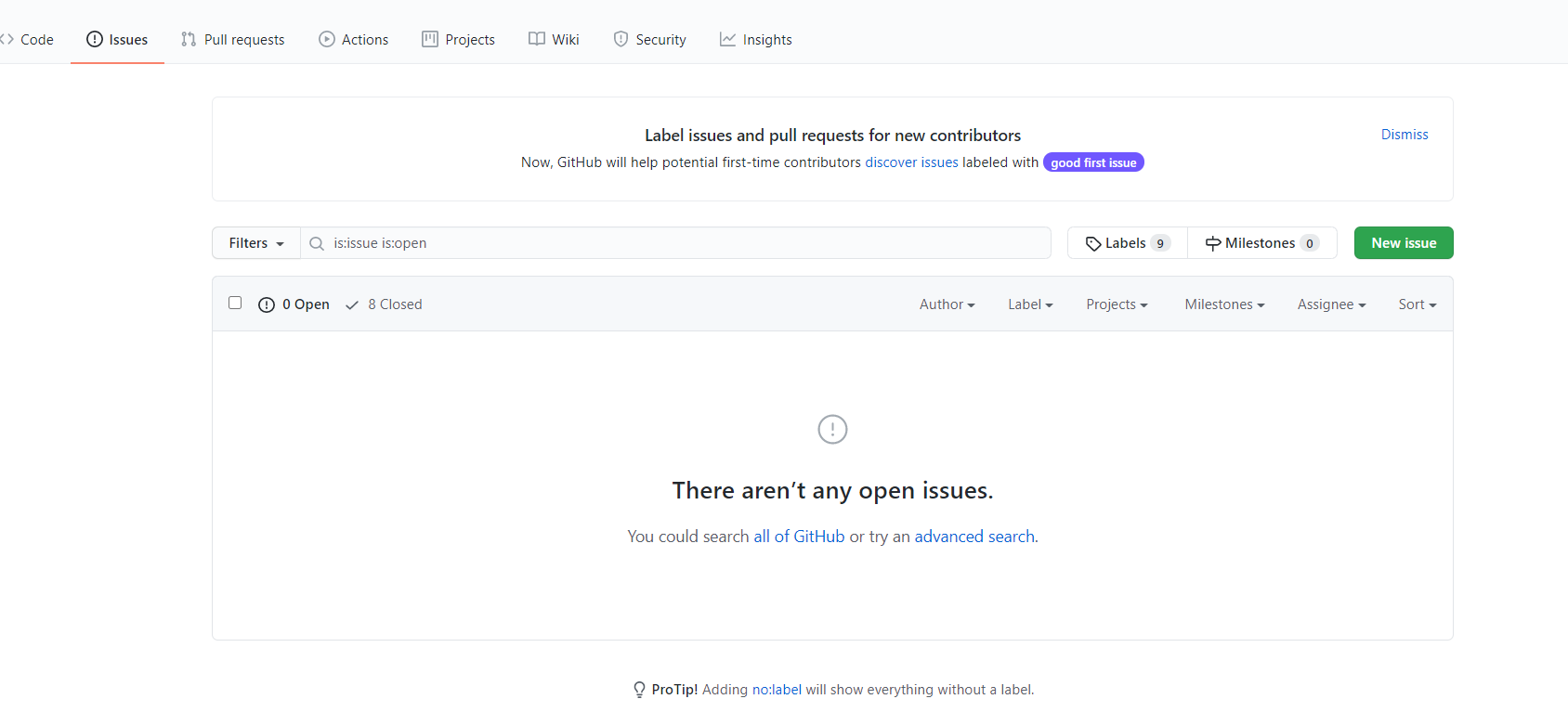


Figure : Git issues

#### Unit Testing

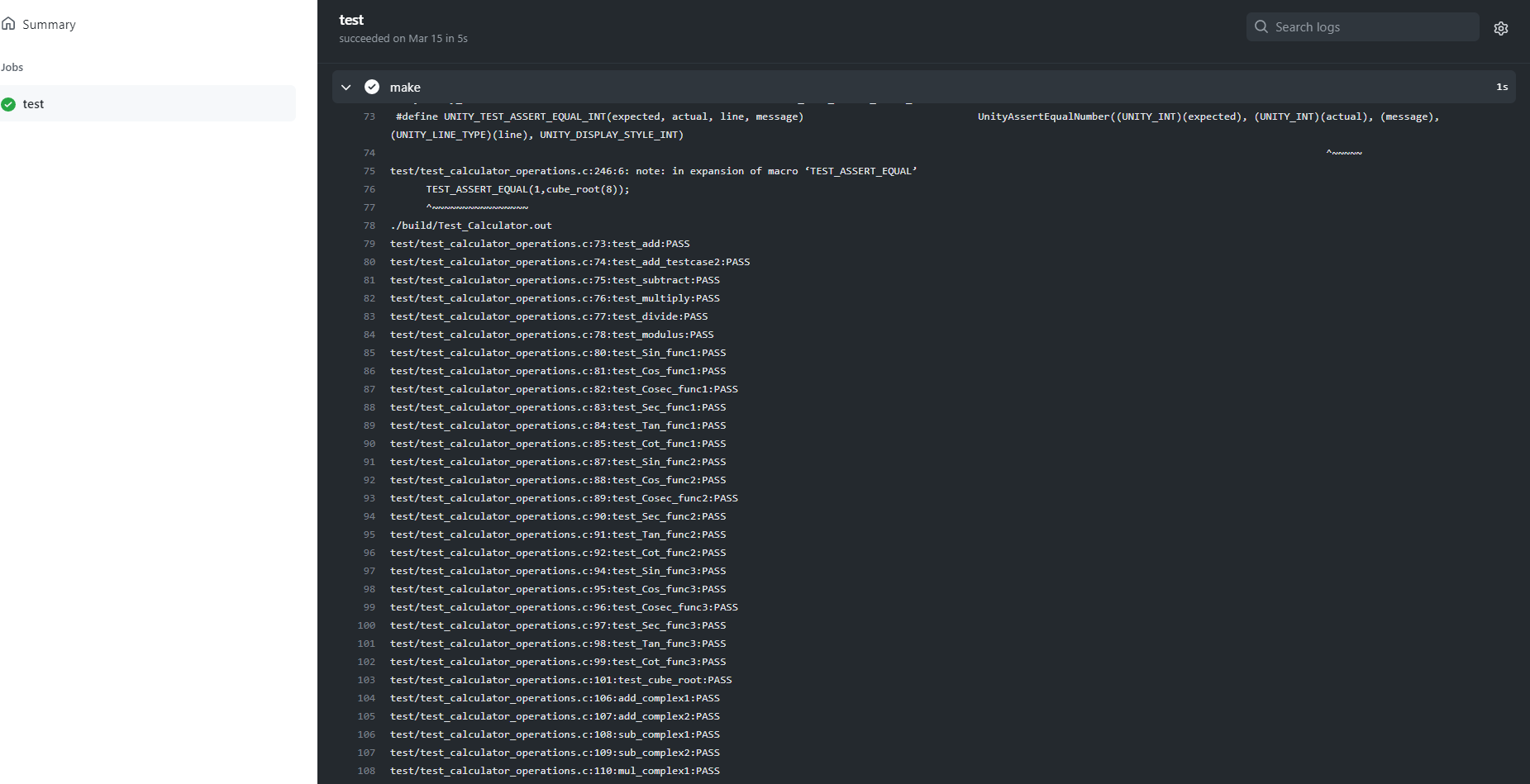


Figure : Unity testing

## **Individual Contribution & Highlights**

My contribution to the project is that I have implemented all arithmetic operations like addition, subtraction, multiplication, division and modulus. I have also implemented a small memory storage operation where the calculator stores the last five results.

To implement the above I did some research work on basic and advance calculators. I have done the SWOT analysis and 4W & 1H analysis to understand the pros and cons of our product. While implementing I have taken care of all the boundary conditions and validated it by writing test cases.

I have written my code in main.c in src folder under implementation. Accordingly, I have added main.h in inc folder under implementation. I have modified the test\_calculator\_operations.c where I have written my test cases.

I have modified Make file so that program should be able to build as well as it is able to perform unity based unit testing.

## **Contributor’s List**

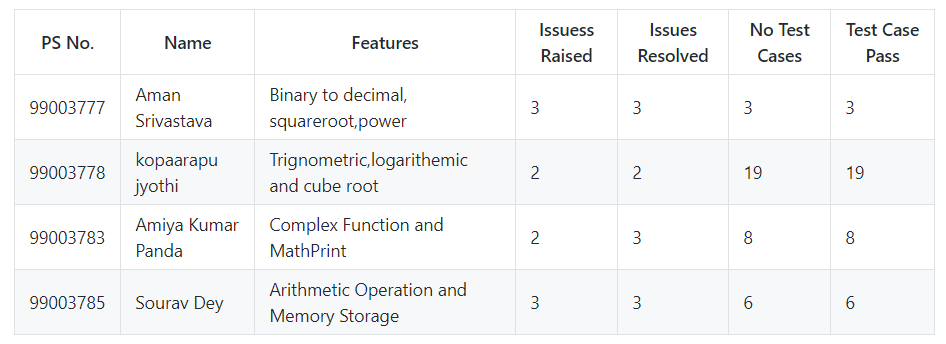


Figure : Contributors list and summary

### Summary

This project was our first step to work in coordination as a team while working towards our development of our individual skills.

Also, we’re unorganized at the beginning but after planning through V-Model we’re able to analyze and design each scenario and perfectly implemented whole project in the given duration. So, we also learned a great skill called time-management.

Technical skills developed: -

* Advanced C programming
* Make File
* Unit Testing through Unity
* Version Control using GitHub

Soft-Skills Developed: -

* Team-work
* Team-management
* Time Management
* Assertiveness

### Challenges faced and how were they overcome

1. At first, we were facing problem with make file later it was overcome by some research work in that topic.
2. Unable to get the test cases passed later some changes have done on test\_calculator\_operations.c file and it shows all the test cases in terminal.
3. Initially, I'm not able to work with make file, I overcome the issue by doing an activity in personal repository.
4. Faced issues during compilation of program and working with GitHub. Our colleagues helped us with clearing of these issues.

### Future Scope (If applicable)

1. Features like matrix operation can be added.
2. Features like AP, GP, AP sum, GP sum can be implemented.
3. Storage Memory can be increased.

### 

# Miniproject -2 Python-> [Individual]

## **Modules Used**

“Module linked to mini project is Advance Python”.

### Topic and Subtopics

The topic is to search a given word from a text file and print the word prior and after the searched word together.

## Objectives & Requirements

High Level Requirements:

|  |  |  |  |
| --- | --- | --- | --- |
| HLR\_ID | HLR\_Requirement | Description | Status |
| HLR\_1 | Text file or input file | Creating a text file from which word is to be searched. | Implemented. |
| HLR\_2 | Printing the searched word. | Printing the searched word in a new file. | Implemented. |

Low Level Requirements:

|  |  |  |  |
| --- | --- | --- | --- |
| LLR\_ID | LLR\_Requirement | Description | Status |
| LLR\_1 | Word to be searched. | The word to be searched must be provided by the user. | Implemented. |
| LLR\_2 | Word search from the text file. | The word to be searched is provided by the user. It is searched through every line in the text file. | Implemented. |
| LLR\_3 | Every instance of the searched word to be printed to a new file. | The searched word along with one word prior and one word after is printed in the new file with file name same as the searched word. | Implemented |

## Design

Python Code:

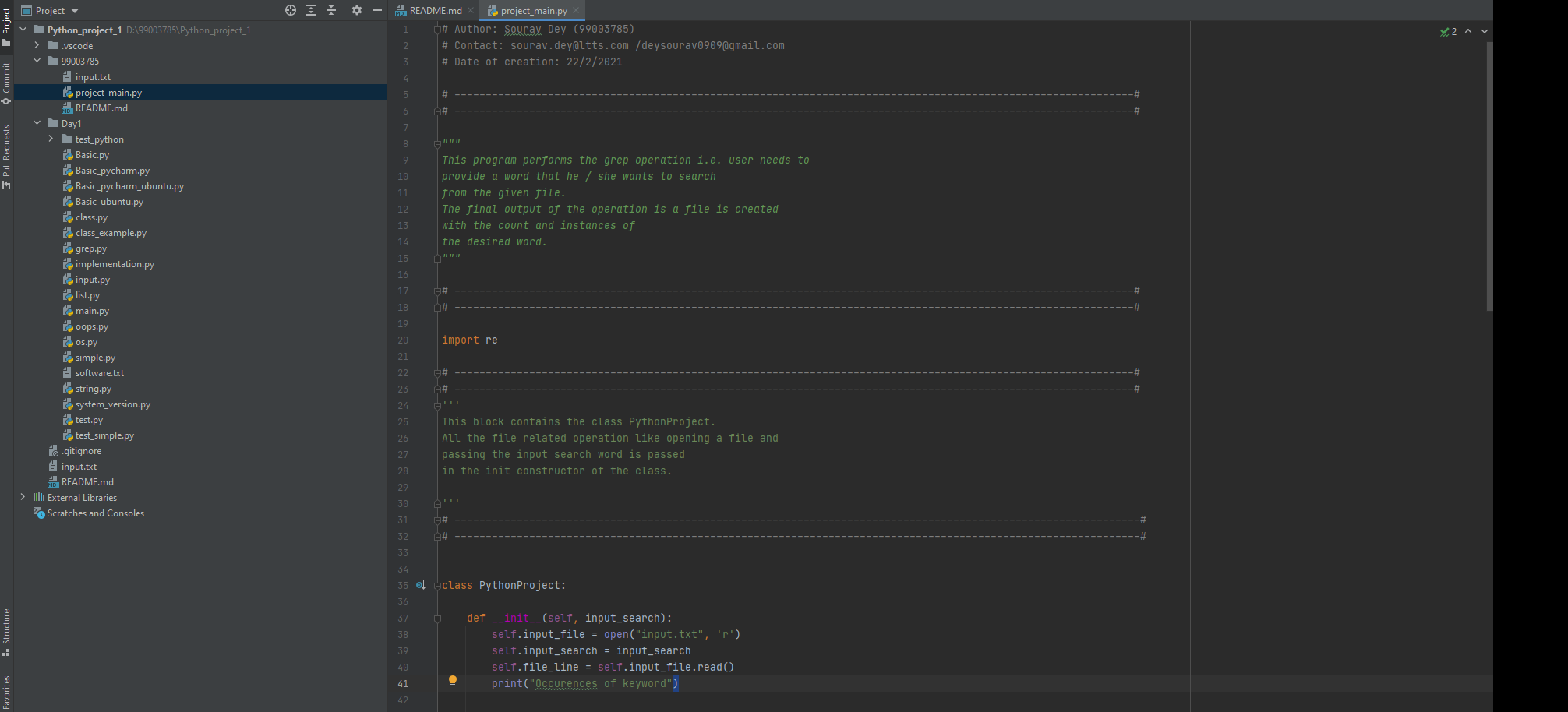


Figure : Code for implementation

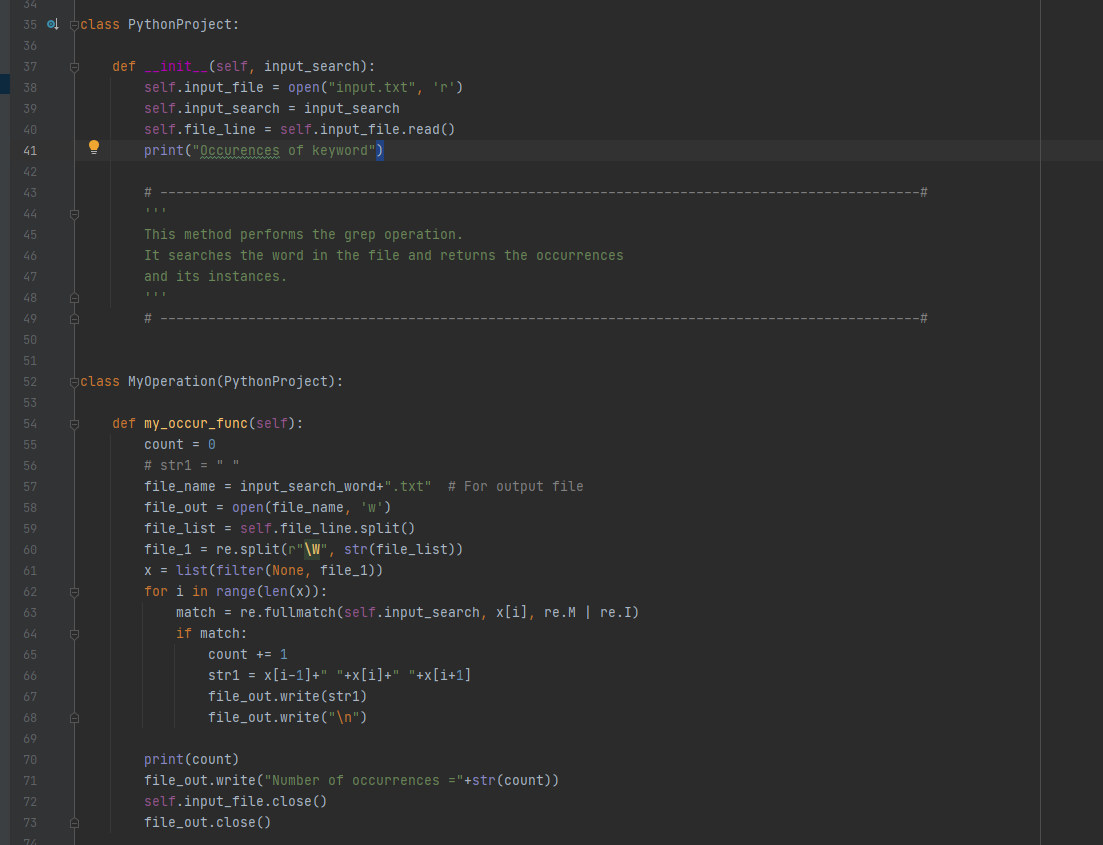


Figure : Continuation of the code.

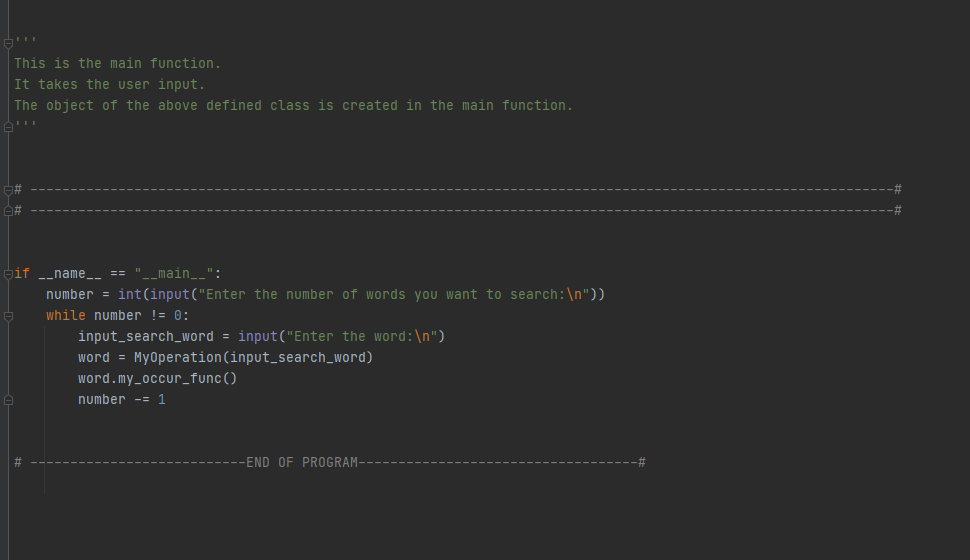


Figure : Continuation of the code.

Output:

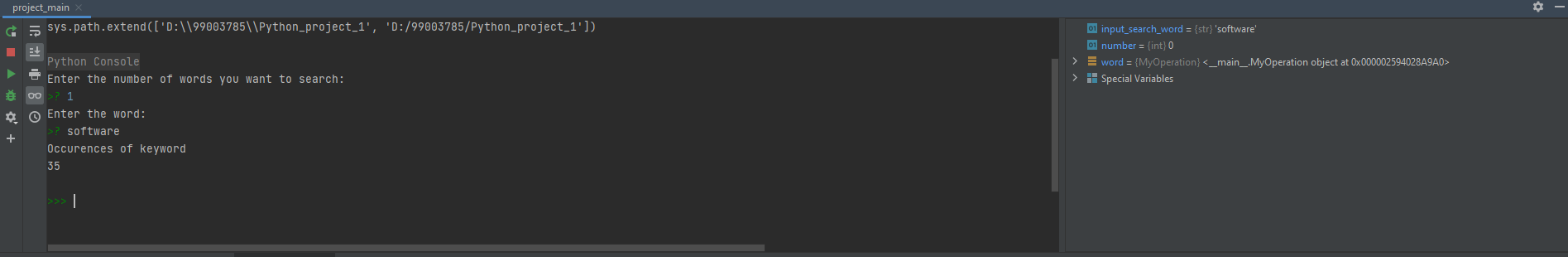


Figure : terminal output

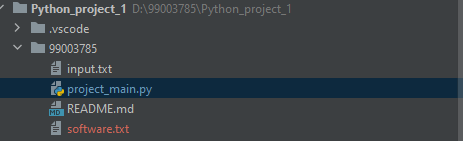


Figure : Creation of new file.

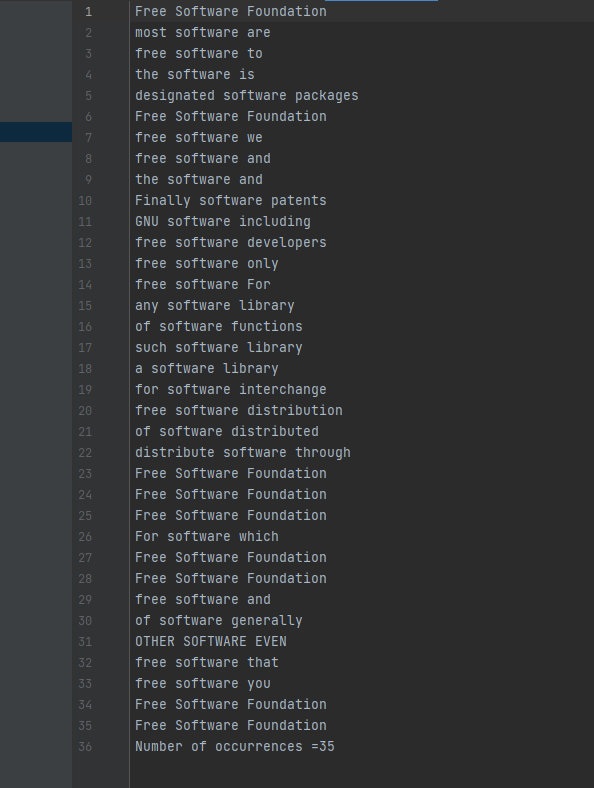


Figure : Output file.

## **Implementation Summary**

The above code will provide the grep operation. User needs to specify the number of words to be searched and the specify the words to be searched.

The code will create different files with word name. Each file will contain all the instances of the word along with one word prior and one word after the searched word.

### Git Link

<https://github.com/99003785/Python_project>

### Git Dashboard

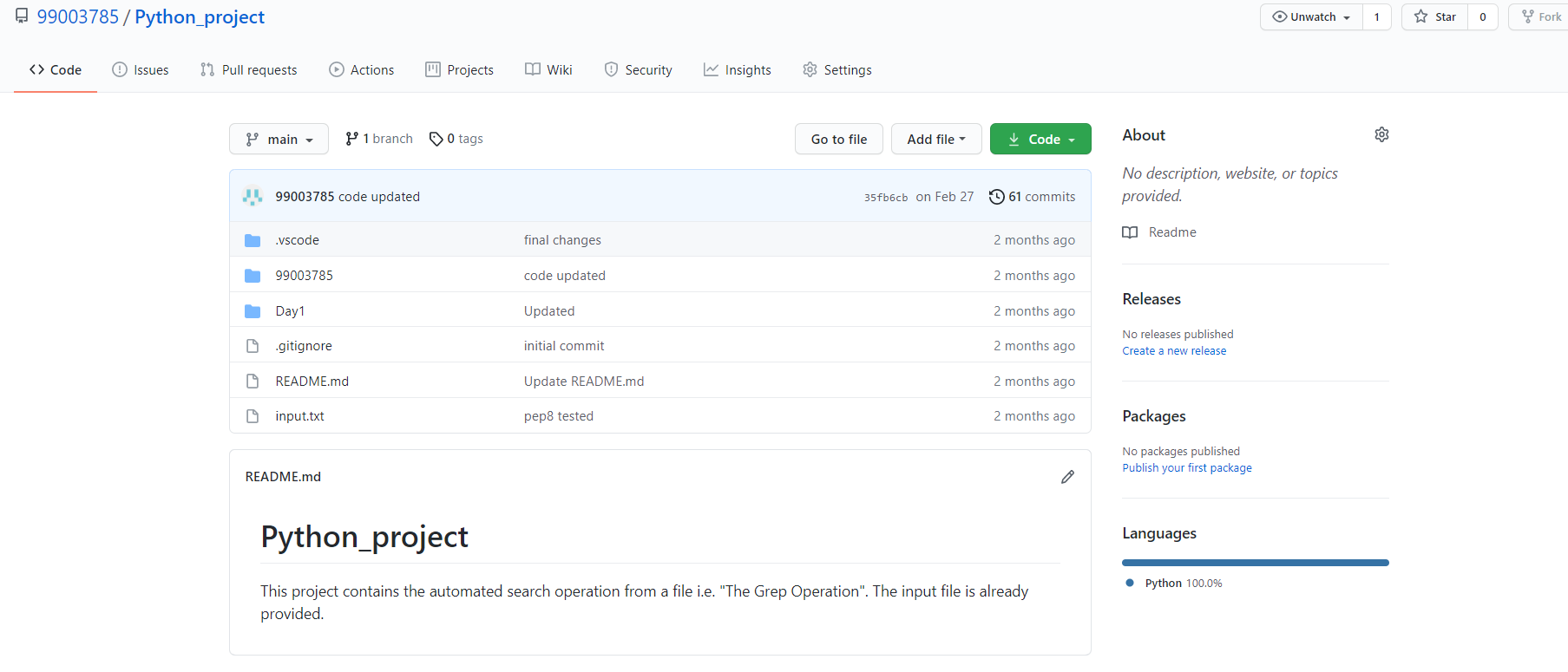


Figure : Git dashboard

### Summary

Word search using automated python scripting is implemented.

### Challenges faced and how were they overcome

1. At first, I faced challenges working with regular expressions, later it was overcome by going through the concepts in depth.

# Miniproject -3 Embedded C-> [Team]

## **Modules Used:**

Modules used in this project are Embedded Systems and Embedded C. These modules were implemented on STM32 board hardware.

### Topic and Subtopics

1)Driver API Development

a) GPIO

b) ADC

c) SPI, UART, I2C

d) External interrupt

e) Debugging using STM Board

2) Driver Development (Hardware Abstraction Level- HAL)

* 1. GPIO
  2. ADC
  3. External Interrupt
  4. Debugging using STM Board

## **Objectives & Requirements**

To implement different Body Control Module (BCM) present in a car using STM32f407VG Microcontroller featuring 32-bit ARM-M4 with FPU core.

The features implemented are as follows:

* Door Lock System
* Interior Lightning
* Seat belt Indicator
* Sun roof control [ Done by me]
* HVAC [ Done by me]
* Wiper System

**Components Used:**

1. STM32f407VG Microcontroller
2. Breadboard
3. LED
4. LDR Sensor
5. Soil Sensor
6. PIR Motion detection sensor
7. RGB Color Sensor
8. Potentiometer Sensor
9. Ring Buzzer Sensor
10. Jumper Wires

Requirements:

High Level Requirements:

|  |  |  |  |
| --- | --- | --- | --- |
| HLR\_ID | Requirement Name | Description | Status |
| HLR\_SRC\_1 | Sun Roof Control System | Implementation of Sun roof control system using switch and LED. | Implemented |
| HLR\_HVAC\_1 | HVAC | Implementation of HVAC using sensors and LEDs. | Implemented |

Low Level Requirements:

|  |  |  |  |
| --- | --- | --- | --- |
| LLR\_ID | Requirement Name | Description | Status |
| LLR\_SRC\_1 | Sun roof control system | When the switch is pressed Red light glows in RGB which indicates opening of sun roof. When the switch is released Red light gets off which indicates sun roof gets closed. | Implemented |
| LLR\_HVAC\_1 | HVAC | Here DHT 11 is used to measure the temperature and humidity. LED gets switched ON when the sensor value exceeds the threshold value. This indicates AC is ON. | Implemented |

## **Design:**

Code:

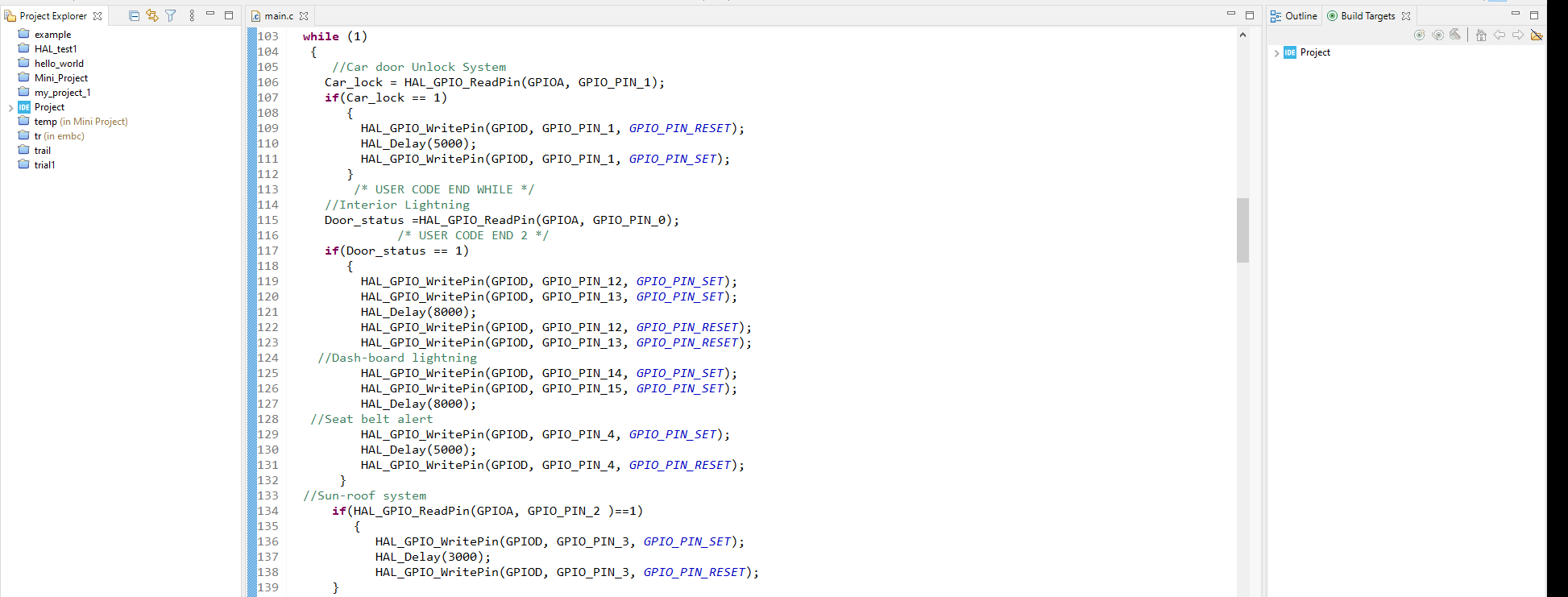


Figure : Code for implementation

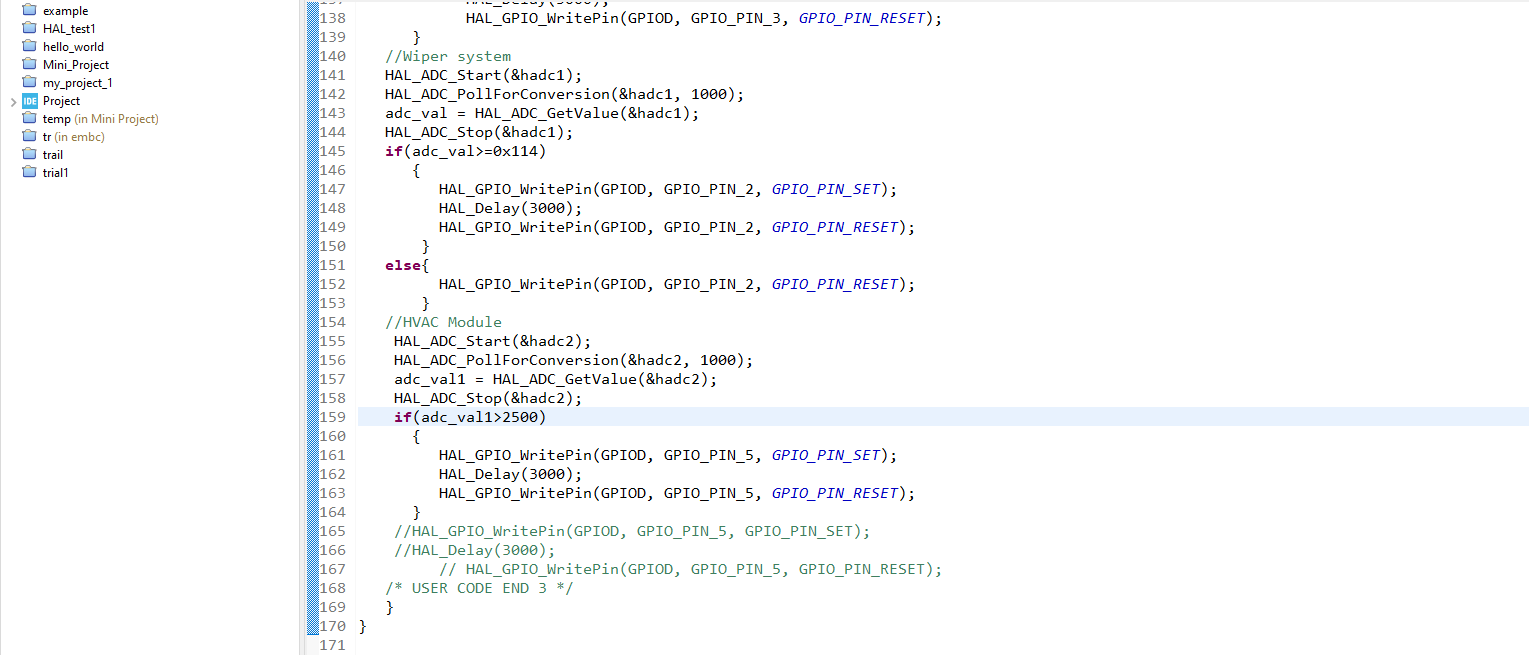


Figure : Code for implementation

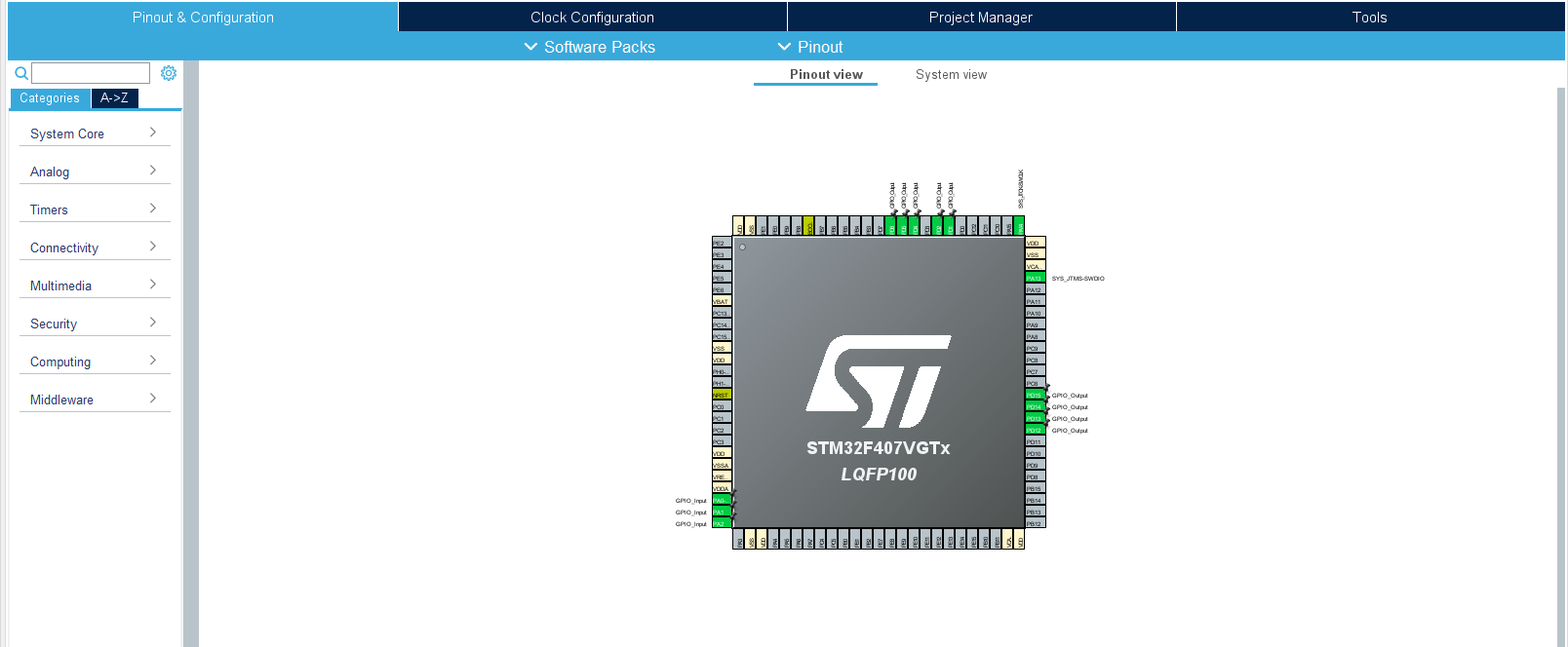


Figure : Port Specification.

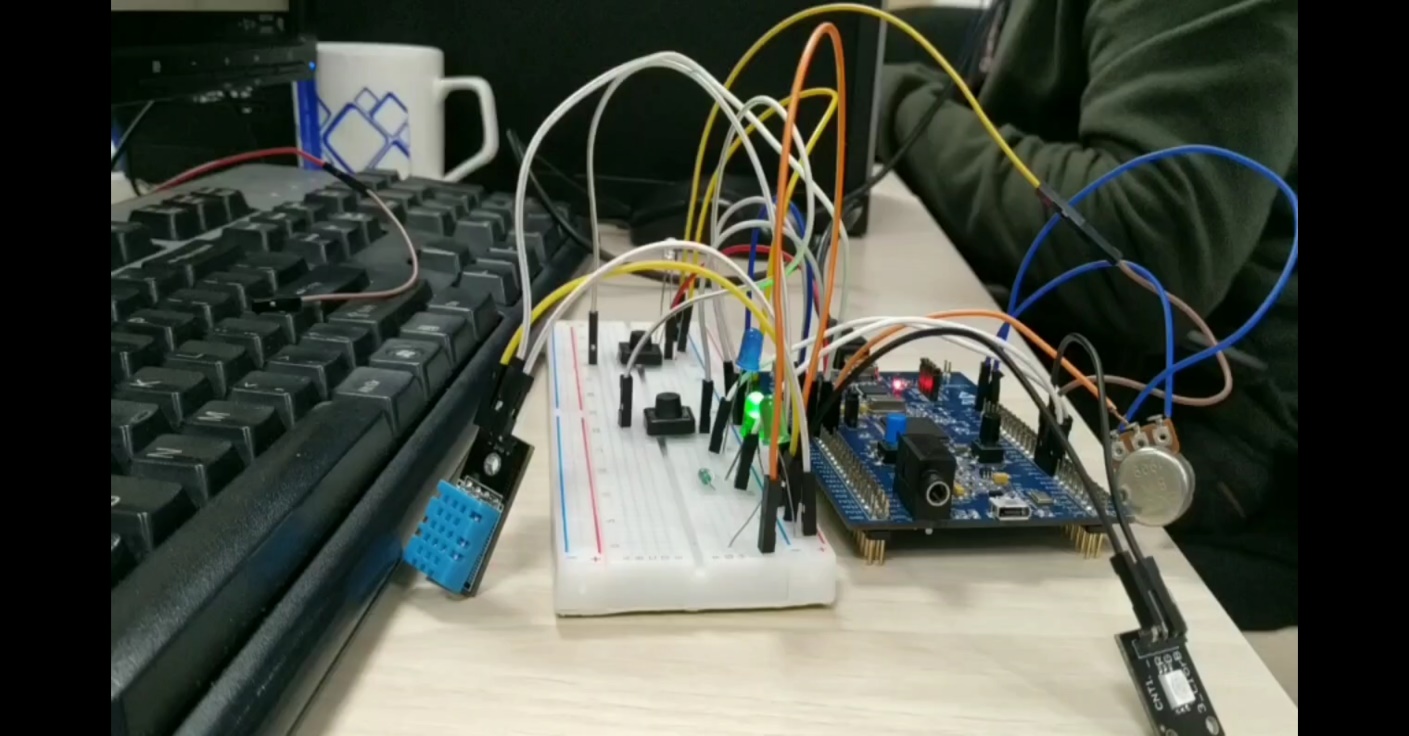


Figure : Hardware circuit along with sensors.

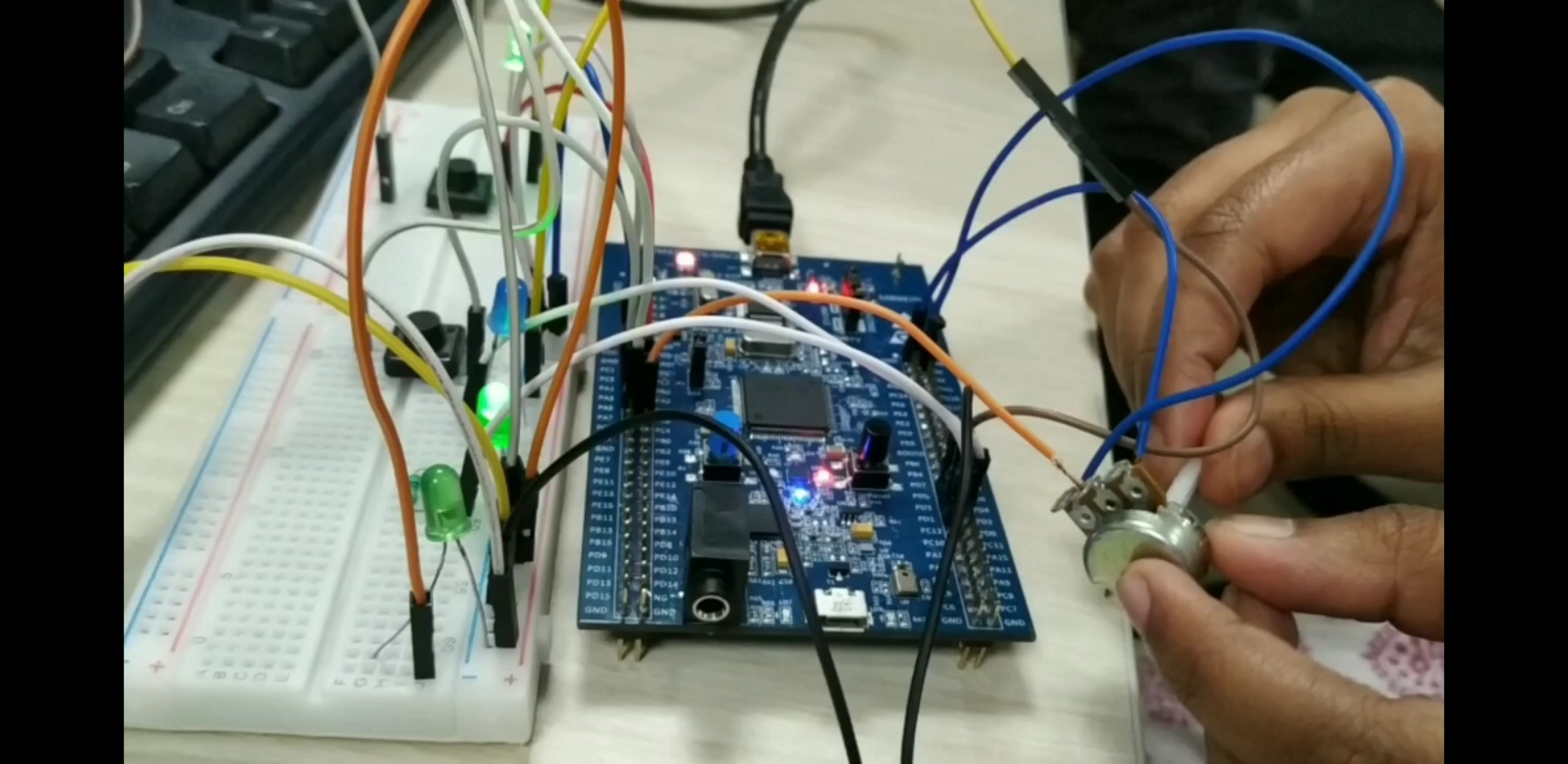


Figure : Hardware circuit implementation.

## **Test Plan**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Test\_ID | Description | Input | Output | Type |
| SRC\_1 | Sun roof control | Pressing of the Push button. | Red Light glows | Requirement type |
| SRC\_2 | Sun roof control | Release of push button. | Red Light gets turned off. | Requirement type |
| HVAC\_1 | HVAC | If the temperature value in DHT11 goes below the threshold value. | LED ON | Requirement type |
| HVAC\_2 | HVAC | If the temperature value in DHT11 goes above the threshold value. | LED OFF | Requirement type |

## **Individual Contribution & Highlights**

My contribution in this project is that I have implemented the Sun roof control module and HVAC module. I have implemented the code in STM32 tool.

The hardware connections were done by using jumper wires and sensors like DHT11. The output response was captured by the LED signals.

### Summary

This project was implemented using STM32f407 board. The code was implemented by creating device drivers and APIs. Sensors like DHT11, smoke sensors were used to implement the respective BCM features. LEDs and buzzers were the main output devices.

### Challenges faced and how were they overcome

* We faced challenges with the hardware components. It was overcome by using alternative methods like printing the value on the screen and other debugging techniques.

# Miniproject -4 MBSE ->[Team]

## **Modules**

Modules used in this project is **MATLAB & Simulink**.

### Topic and Subtopics

1. MATLAB
2. Simulink

## **Objectives & Requirements**

Objective:

To implement different Body Control Module (BCM) feature present in a car using MATLAB and Simulink modelling. To test the respective feature and whole integrated system using Manual testing and Automating testing.

Here GMC Seirra truck is taken as a reference to build the models.

The features implemented are as follows:

* Hood Release [Done by me]
* Manual HVAC module [Done by me]
* Airbag
* Door Lock module
* Wiper Control Module
* Tail Gate release
* Interior Lights
* Exterior Lights
* Seat belt module
* Sun roof

Requirements:

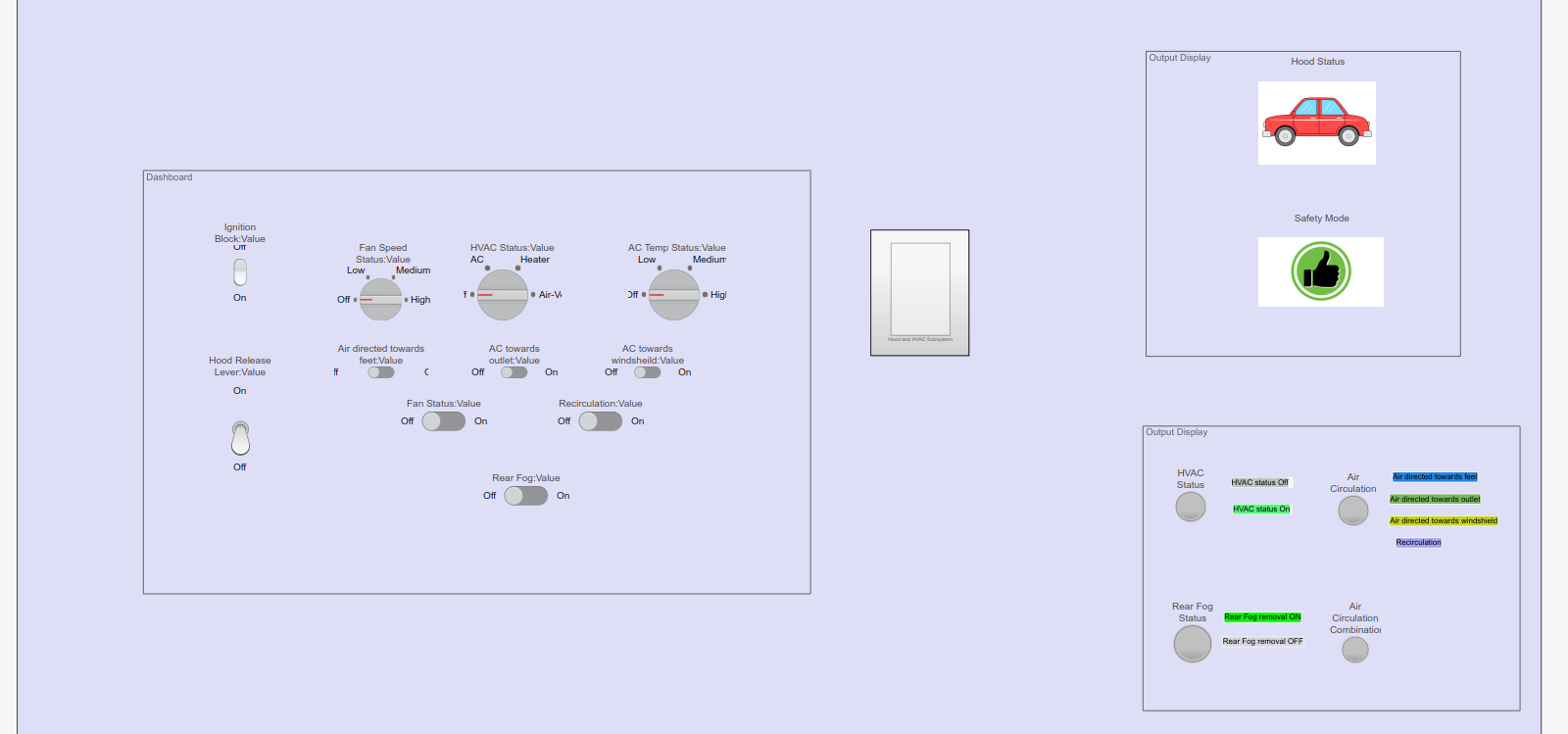
High Level Requirements:

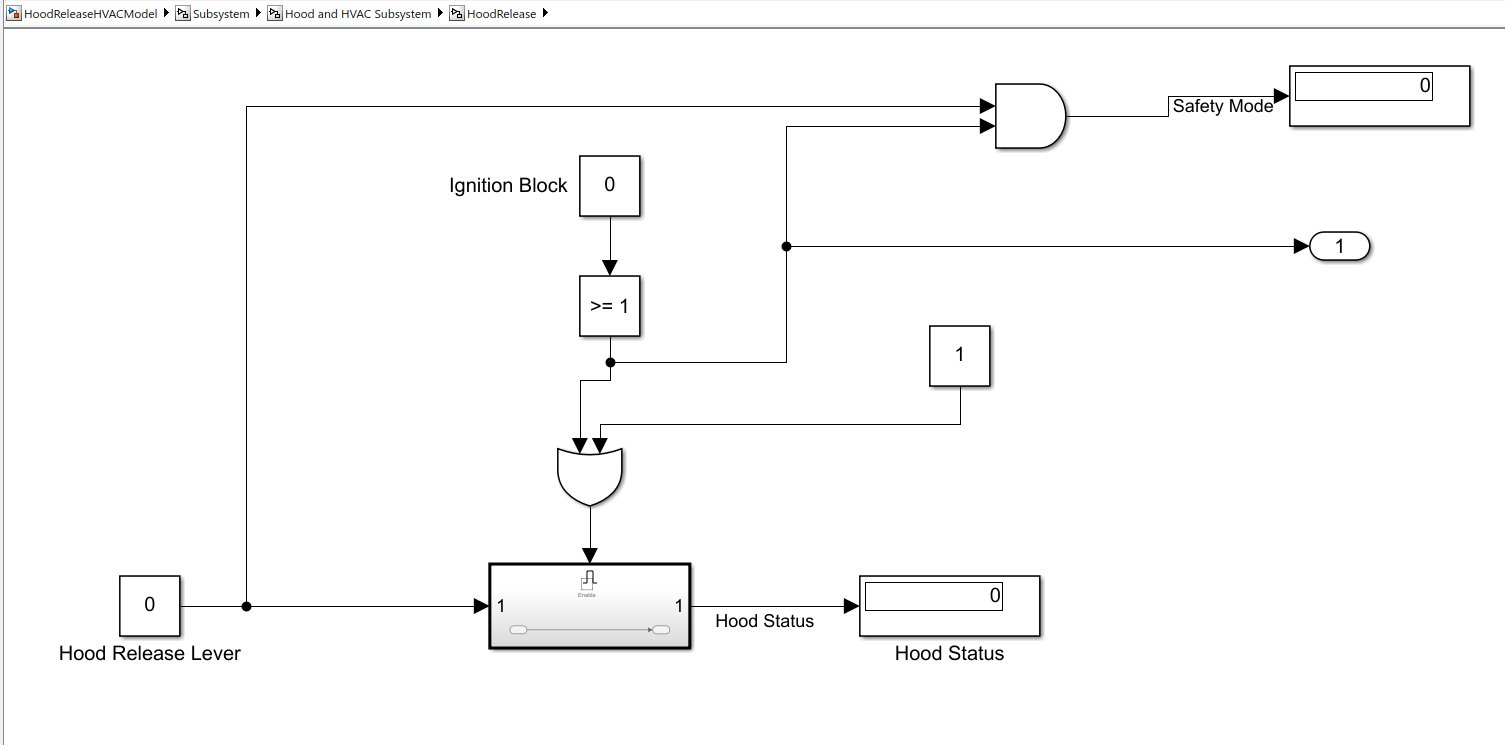
1. Hood must open when the button is switched ON. Hood must close when the button is switched OFF.
2. HVAC should operate according to the inputs given by the driver or the passenger
3. For HVAC to work fan should be switched ON.
4. Rear Fog removal.

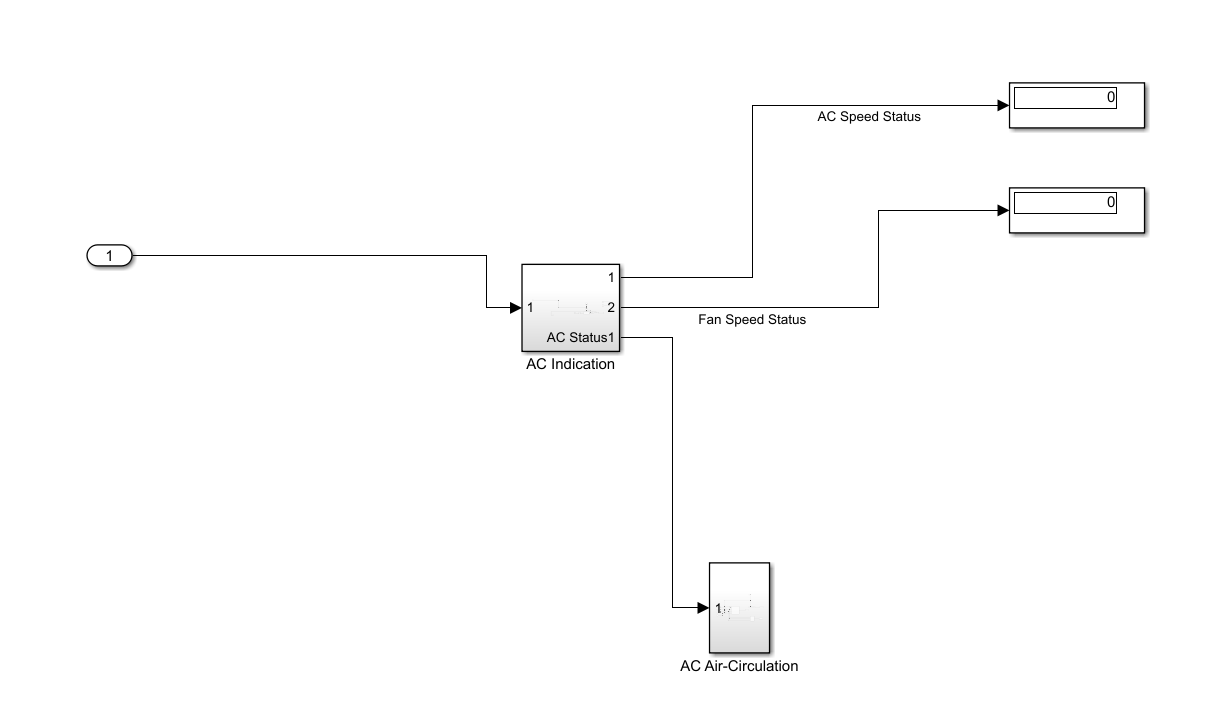
Low Level Requirements:

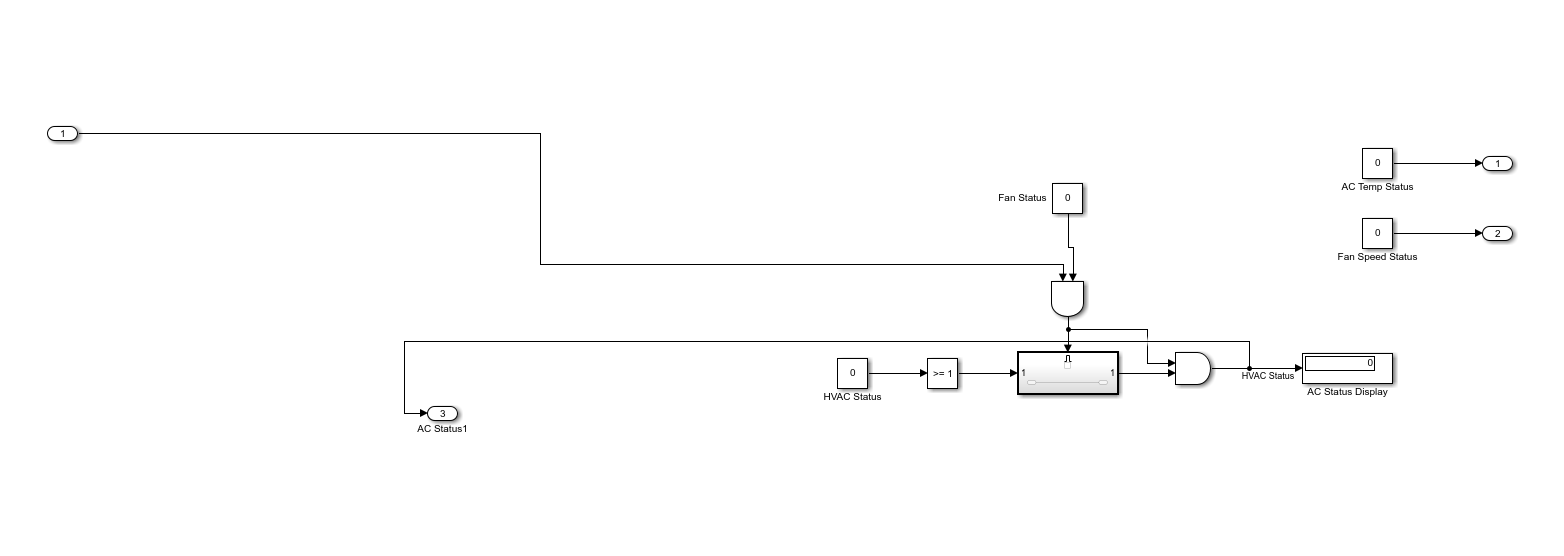
1. Hood must open when switch is ON and hood must close when switch is OFF.
2. When the AC is switched ON different modes of air control can be implemented by the passenger or the driver.
3. When the AC is switched ON different combination of the air control can be implemented.
4. When rear fog removal is implemented humid less air is blown to the rear mirror. Fan speed is also high.

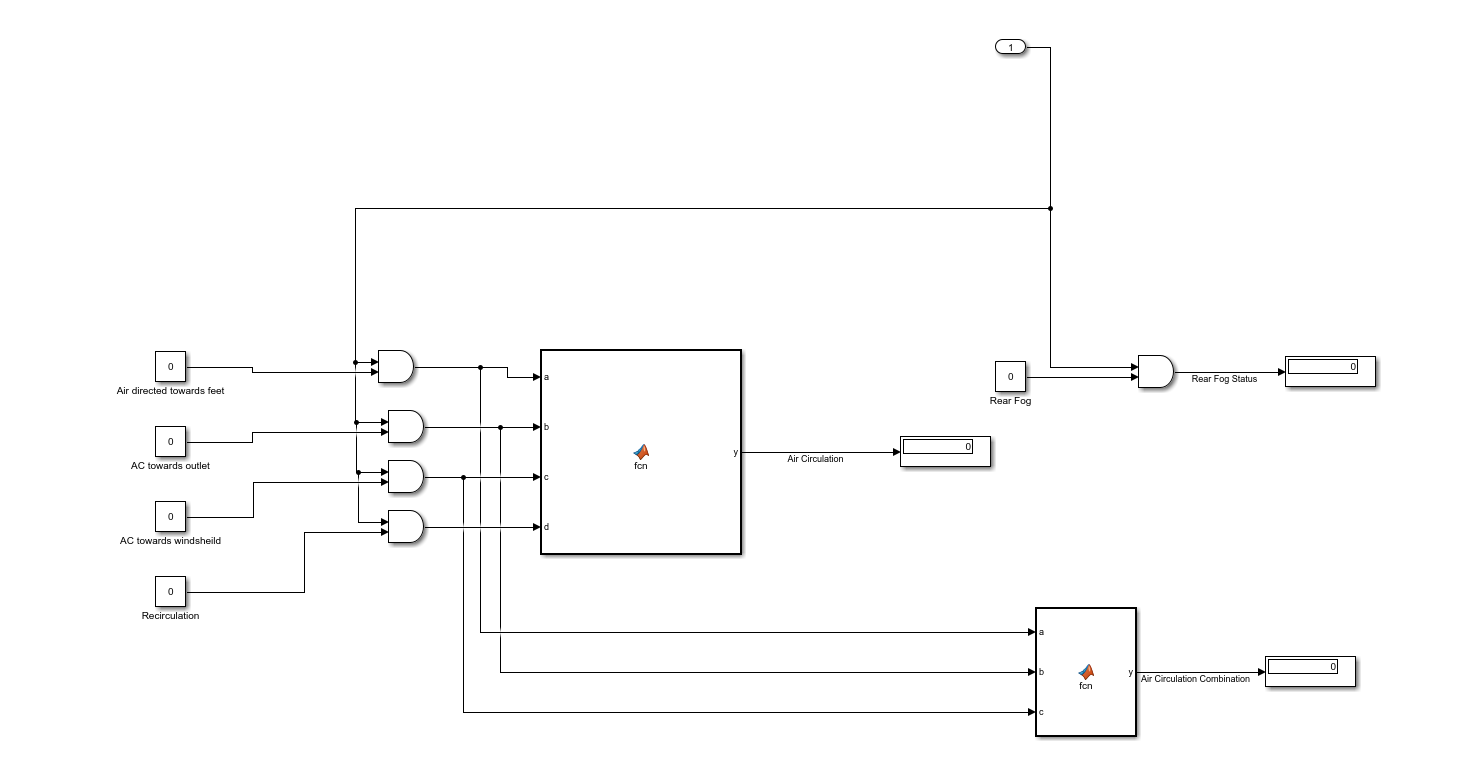
## **Design**

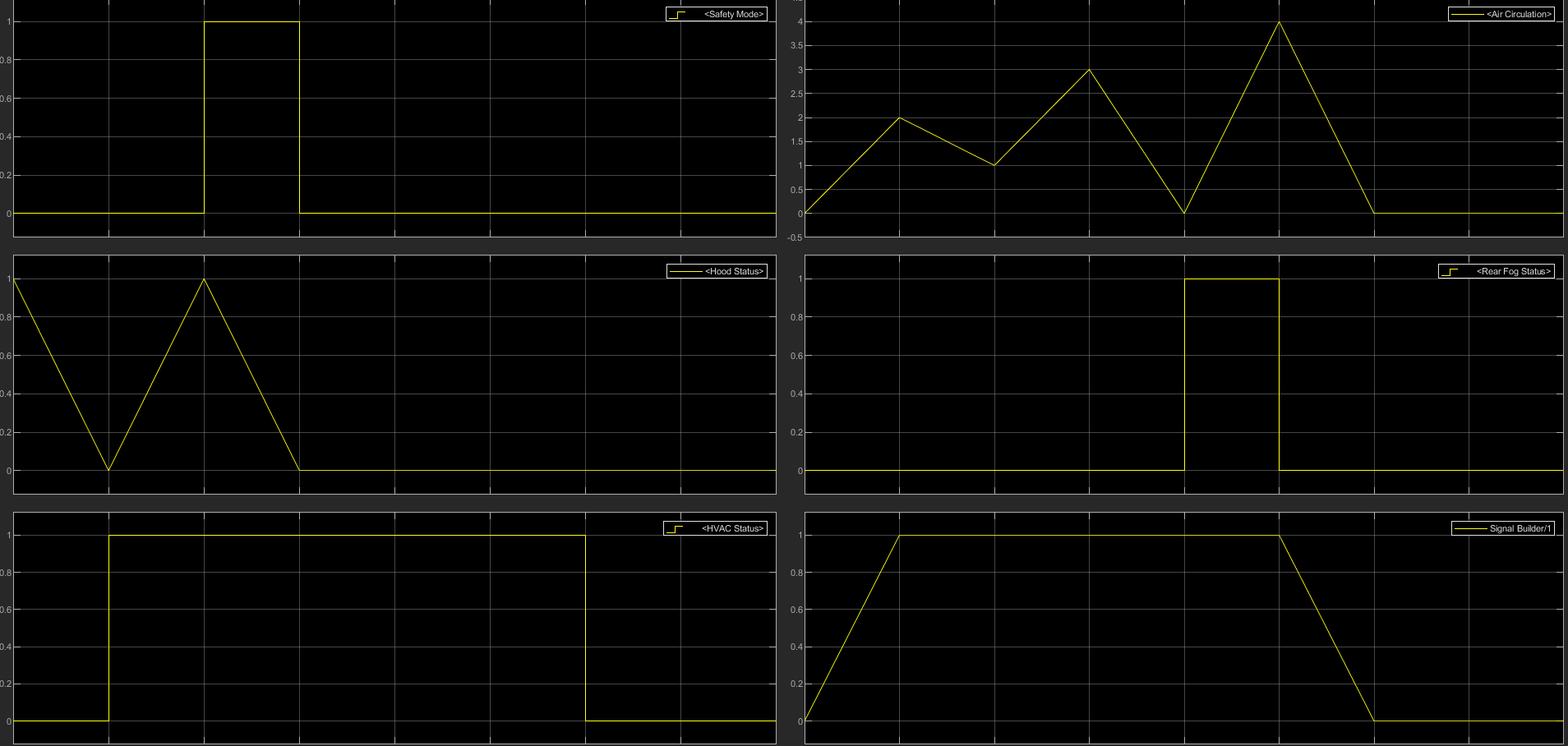


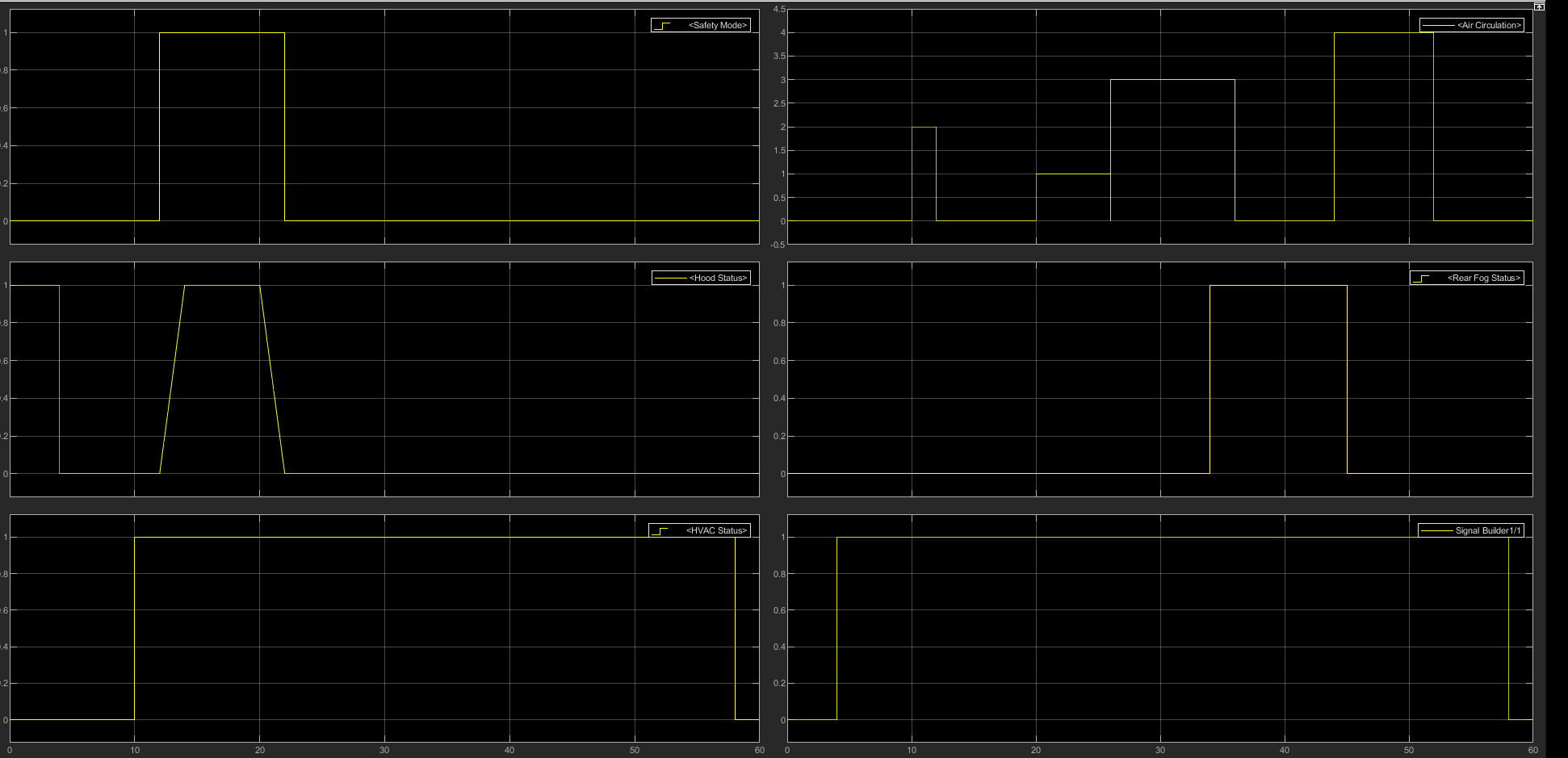












## Test Plan

## Individual Contribution & Highlights

|  |  |
| --- | --- |
| SL NO. | Role in Project |
| 1. | Designing of the subsystems – HVAC module and Hood Release Module. |
| 2. | Gathering requirements and designing all corner cases for automated testing using signal builder. |

## **Implementation Summary**

Designing of the BCM features i.e. HVAC module and Hood release using MATLAB and Simulink. Designing of the manual testing model and automated testing models. Excel sheet is created to validate all the test case and the corner cases.

Integration of all the features and performing integration testing using signal builder and excel sheet.

### Summary

### Challenges faced and how were they overcome

# Miniproject -2 [Team/Individual]

## Module/s

### Topic and Subtopics

## Objectives & Requirements

## Design

## Test Plan

## Implementation Summary

### Git Link

### Git Dashboard

### Summary

#### Git inspector summary

#### Build

#### Code quality

#### Unit Testing

#### Issues

## Individual Contribution & Highlights

### Summary

### Challenges faced and how were they overcome

# Miniproject -2 [Team/Individual]

## Module/s

### Topic and Subtopics

## Objectives & Requirements

## Design

## Test Plan

## Implementation Summary

### Git Link

### Git Dashboard

### Summary

#### Git inspector summary

#### Build

#### Code quality

#### Unit Testing

#### Issues

## Individual Contribution & Highlights

### Summary

### Challenges faced and how were they overcome